UNIVERSITY OF TARTU
Institute of Computer Science
Cyber Security Curriculum

Raul Nugis

Forensic Data Properties of Digital Signature BDOC and ASiC-E Files on Classic Disk Drives

Master's Thesis (30 ECTS)

Supervisors: Pavel Laptev
Raimundas Matulevičius

Tartu 2018

Forensic Data Properties of Digital Signature BDOC and ASiC-E Files on Classic Disk Drives

Abstract:

This thesis reviews the contents and observes certain properties of digitally signed documents of BDOC and ASiC-E container formats. After reviewing a set of sample containers, the author comes up with a header and footer combination (signature) significantly improving pin-pointed carving-based recovery of those files from a deleted state on NTFS formatted uncompressed volumes in contiguous clusters, taking into account the geometry of classic disk drives. The author also describes forensically meaningful attributive data found in ZIP Headers and Central Directory, XML signatures as well as embedded ASN.1 encoded data of the sample files and suggests an algorithm for the extraction of such data. Based on these findings, the author creates scripts in Python and executes a series of tests for file carving and extraction of attributive data. These tests are run over the samples placed into unallocated clusters and the results are compared to several mainstream commercial forensic examination suites as well as some popular data recovery tools. Finally, the author web-scrapes a large number of real-life documents from a government agency's public document registry. The carving signature and the data-extractive algorithm are thereafter applied on a larger scale and in an environment competitively supplemented with structurally similar containers.

Keywords:

Attribution, file carving, file signature, file header, file footer, digitally signed document, ZIP Local File Header, ZIP Central Directory Record, ZIP Central Directory End Record, XML signature, ASN.1 encoded object, Web-Scraping, BDOC, ASiC-E

CERCS: P170, Computer science, numerical analysis, systems, control

Digitaalselt allkirjastatud BDOC ja ASiC-E failide kohtuekspertiisis huvipakkuvad omadused klassikalistel kõvaketastel

Lühikokkuvõte:

Käesolevas magistritöös vaadeldakse BDOC ja ASiC-E digitaalselt allkirjastatud dokumendikonteinerite sisu ning kirjeldatakse nende huvipakkuvaid omadusi. Teatava hulga näidiskonteinerite vaatlemise järel pakub autor välja faili päise ja faili jaluse kombinatsiooni (signatuuri), mis oluliselt parandab nimetatud failide kustutatud olekust sihitud taastamist külgnevatest klastritest NTFS vormindatud tihendamata kettal, võttes arvesse klassikalise kõvaketta geomeetriat. Ühtlasi kirjeldab autor kohtuekspertiisi koha pealt tähendust omavaid andmeid ZIP kohaliku faili päises ja keskkataloogi kirjes, XML signatuuris ja ASN.1 kodeeritud kihtides ning nende kättesaamise algoritmi. Nendele järeldustele tuginedes loob autor Püütoni skripte ja viib läbi mitmeid teste failide taastamiseks faili signatuuri järgi ning huvipakkuvate andmete väljavõtmiseks. Teste viiakse läbi teatava valiku failide üle ja tulemusi võrreldakse mitme kohtuekspertiisis laialt kasutatava peavoolu töökeskkonnaga, samuti mõningate andmetaaste tööriistadega. Lõpuks testitakse magistritöö käigus pakutud digitaalselt allkirjastatud dokumentide taastamiseks mõeldud signatuuri ja andmete väljavõtmise algoritmi suurel hulgal avalikust dokumendiregistrist pärit kehtivate dokumentidega, mis saadi kätte spetsiaalselt selleks kirjutatud veebirobotiga. Nimetatud teste viiakse läbi dokumentide üle, mille hulgas on nii digitaalselt allkirjastatud dokumente kui ka teisi, nendega struktuurilt sarnaseid dokumente.

Võtmesõnad:

Tõendusmaterjali sidumine isikuga, faili taastamine signatuuri alusel, faili signatuur, faili päis, faili jalus, digitaalselt allkirjastatud dokument, ZIP kohaliku faili päis, ZIP keskkataloogi kirje, ZIP keskkataloogi lõpukirje, XML signatuur, ASN.1 kodeeritud objekt, veebirobot, BDOC, ASiC-E

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Acknowledgements

The author wishes to thank the Estonian Competition Authority, and the Authority's Deputy Director-General Kristel Rõõmusaar, for her warm support and inspiration for the author's studies in digital forensics.

The author wishes to thank the Estonian Forensic Institute's digital forensics experts, Oliver Olt and Aivo Vispert, for their valuable insights into the forensic issues associated with the subjects of this thesis.

Table of Contents

1 Dictionary

In this thesis the following notations are used.

Digitally Signed Document (DSD) is a file (digital document) together with an associated XML signature, stored in a container of the BDOC or ASICE types. Legally, DSDs are regulated by EU and national legal acts such as, for example, § 24 (1) of "*Electronic Identification and Trust Services for Electronic Transactions Act*" (RT I 2000, 26, 150)[1] and conditions set in § 24 (2) articles 1-4 of the same act.

Container is a file compliant with the technical specifications of the ZIP File Format [1]. In addition to ZIP, a container may also comply with additional specifications such as those of ASIC [14]. In filesystems that allow file extensions, containers may have extensions including '.zip', '.asice', '.docx', etc.

Outer (ZIP) Layer is container's Local File Headers, Data Descriptors, Central Directory Records and Central Directory End Record as defined in ZIP specifications [1].

Intermediate (XML) Layer is the XML formatted contents of an XML signature file in a DSD container compliant with XAdES specifications [19].

Inner (ASN.1) Layer is the contents of 'X509Certificate', 'EncapsulatedX509Certificate', 'EncapsulatedOCSPValue' and, 'EncapsulatedTimeStamp' objects inside an XML signature file, compliant with ITU ASN.1 encoding standards [46].

XML Signature is a file with the naming scheme 'signature*.xml' holding a signature or signatures associated with a signed file (document), compliant to XAdES specifications [19] and stored inside a container.

Digital Forensic Examination (DFE) generally includes acquisition of data from a source, analysis of the data and extraction of evidence, as well as preservation and presentation of the evidence [2]. This thesis concentrates on the acquisition of forensic images of media, data recovery in the form of file carving and extraction of attributive data from carved files on NTFS formatted classic disk media.

File Carving is part of DFE and reconstructs files based on their contents, rather than using metadata that points to the content [3], making use of the file header (start of the file) as well as the file footer (end of the file) [4] or other means to identify the end of the file, and data in between. In this thesis, only clustered, contiguous data carving is referred to. The file header and file footer, represented in certain way, can be called the file signature and must not be confused with an XML signature of DSD, defined above.

Attributive data is any data that can be extracted from DSD during DFE and is helpful in learning about the signer or their environment. The extraction of attributive data is important in forensics because attribution is one of its principal tasks [5, 25].

[1] Consolidated English translation available at https://www.riigiteataja.ee/en/eli/527102016001/consolide, retrieved on 14.03.2018.

2 Introduction

The legal framework enabling electronic signatures in the EU and in Estonia appeared almost at the same time, nearly twenty years ago. Since its adoption, the Estonian ID-card based digital signing has undergone rapid expansion and by the middle of the last decade had almost completely permeated both the public and private sector, forming an integral part of documentation and archiving. While widely accepted by the general public, these documents pose certain challenges from the point of view of digital forensics. This is mainly due to the similarity of digitally signed documents' containers to other ZIP files, the shortcomings of known signatures in separating different subtypes of containers and the difficulties of penetrating the multi-layered structure of digitally signed documents by mainstream indexing and keyword searching tools.

Lack of support from major commercial forensic solutions and a shortage of forensically comprehensive descriptive sources has motivated the author to undertake this work in order to explore the internals of digitally signed documents and explain what is learnt. As a result of this exploration, the author would like to come out with a signature capable of recognising digitally signed documents and extracting them from raw data. The author also undertakes the observation and description of pieces of data found in the observed digitally signed documents, which, in the author's view, hold attributive value, that is, are helpful for a forensic examiner in learning more about the signer and their environment, including, but not limited to, the signer's data inside a signature-embedded certificate.

3 Background

This chapter looks at the goals set by the creators of the digital signature's framework and how it was developed over time, as well as the impact that digitally signed documents have on digital forensic investigations today. The chapter also highlights certain elements specified in the standards governing ASICE and BDOC containers, which may be relevant for the purpose of this thesis.

3.1 The Road to Digitally Signed Documents

Even though Estonia's digital signature supporting website www.id.ee has been registered to SK ID Solutions AS since 04.07.2010[2], a website was first opened at this domain in 1998[3] and in 1999 the concept of the ID-card was revealed. In the words of the authors' of the original website the purpose of the project was to develop a new personal identification card that would be a generally acceptable identification document and contain both visually and electronically accessible information. The envisaged ID-card was planned as multifunctional, enabling personal identification and containing a personal digital signature. At that time, the number of personal computers in people's homes was estimated to be about 40 – 60 thousand [6], which meant that roughly 80-90% of ID-cards, once issued to all citizens, would rarely be used. The authors of the concept behind the ID-card were clearly investing in the future, not in the present. This might be one of the reasons why by November 2001 the file format of the documents that could be signed digitally using the ID-card had yet to be planned. A proposal was made [7] that the format should be XML.

In these early days, a digitally signed document (DSD) was envisaged as information recorded on any type of media, which is created as result of the activity of an organisation or person, and whose contents, form and structure is sufficiently provable. In the view of the authors, DSD would have an additional metapart attached to it, which would enable verification of the document. A DSD's metapart is a digital signature and time stamp, which connect the document to its author and creation time [8] in an undisputable way. Today, the latter concept is usually understood in English as non-repudiation [50].

At about the same time as the events unfolding in Estonia, the European Communities, predecessor of today's European Union, adopted "*Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures*" (further referred simply as "the Directive"[4]), later repealed by the so-called eIDAS Regulation[5]. The Directive defined, in article 1, paragraphs 1 and 2, an advanced electronic signature in terms of identification, which was achievable by uniquely linking the signature to the signer and capable of identifying the signer; authentication by creating a signature using means that the signer can maintain under their sole control; and integrity and verifiability, which was made possible by ensuring that the signature is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable. The Directive also raised this type of electronic signature to the same legal level as a traditional signature.

[2] Domain records at www.internet.ee, retrieved on 13.01.2018.
[3] Old website available at Wayback Machine https://web.archive.org/web/19981201000000*/www.id.ee, retrieved on 13.01.2018.
[4] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:31999L0093, retrieved on 14.03.2018.
[5] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv%3AOJ.L_.2014.257.01.0073.01.ENG, retrieved on 14.03.2018.

The Estonian Digital Signature Act (2000, repealed in 2016)[6] defined, in § 2, the term "digital signature" in a somewhat different fashion. According to the definition[7], a digital signature is a data unit, created using a system of technical and organisational means, which is used by a signer to indicate their link to a document. A digital signature is created by using the data necessary for giving a signature contained in a secure signature creation device (private key) to which the data needed for verification of the signature contained in a signature verification device (public key) uniquely corresponds. The act stipulated that a digital signature and the system of using the digital signature must have the following properties:

- enabling unique identification of the person in whose name the signature is given;
- enabling determination of the time when the signature is given;
- enabling linking the digital signature to data in such a manner as to preclude the possibility of changing the data, or the meaning thereof, undetectably after the signature is given.

The core national definition of "digital signature" is therefore different from the one originally established in the EU due to the former emphasising identification of the time of signing.

Over the following years, Estonian ID card-based solutions, including document signing, were spreading like wildfire and by the middle of the first decade of the century the private sector was actively switching over to digital signatures, the process of which was branded "paperless" at that time [9,10]. The EU-wide electronic signature's legal framework had not taken off at the same rate [11].

Today, the European Union recognises, format-wise, 3 relevant electronic signature formats[8] and one container format, of which the container format (ASiC, with its sub-version ASiC-E) is relevant to this thesis. The Estonian container format BDOC is declared to be fully compliant with the ASiC standard[9]. A plenitude of other digital signing solutions exist across the Member States, which do not necessarily stick to the Union standards [12]. It is difficult to know with any degree of certainty what formats are used for DSDs in each individual Member State, or whether their signatures and containers are recognised as EU compliant. One piece of research, for example, outlines some of the e-document and signature formats used in different Member States in 2015. The researchers identified that document format specifications were nationally adopted in only 5 Member States, of which only one used the EU backed ASIC container [13].

Even though it is difficult to soundly ascertain how popular DSDs are EU-wide, and what role they play in public administration and business, they are undoubtedly widespread in Estonia. Estonia's Health Authority's public document registry search[10] results, as conducted by the author in March 2018, suggest registration of 5532 PDF documents, 4172 BDOC documents, 425 DDOC documents, 280 RTF documents, 280 DOCX documents, 102 ZIP archives, and 11 ASICE documents. This makes DSDs second only to PDFs, at least when public administration is concerned, suggesting that DSDs must play an important role in DFEs concentrating on the extraction of evidentiary information from documents.

---

[6] Consolidated English translation available at https://www.riigiteataja.ee/en/eli/ee/Riigikogu/act/508072014007/consolide, retrieved on 23.02.2018.

[7] Text based on later translation.

[8] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/e-Signature+standards, retrieved on 13.01.2018.

[9] https://www.id.ee/?lang=en&id=34336, retrieved on 14.03.2018.

[10] http://dokumendihaldus.terviseamet.ee/default.aspx, retrieved on 14.03.2018.

3.2 Digital Forensic Implications of DSDs

Digital signatures were originally meant for business. The 1999 EC Directive's recitals[11] 4, 10, 17, 19, 23, 24 explain the necessity of what is sometimes more narrowly defined as a digital signature, more broadly referred to as an electronic signature, in terms of trade, commerce, contracts, public administration, public procurements, taxation, social security, health and justice systems. Similarly, the 2014 Regulation, which replaced the Directive, referred to public and private online services, electronic business and electronic commerce as the relevant areas. Similar considerations are repeated in the introductory part of the EU's standards developed and published by the European Telecommunications Standards Institute (ETSI) as well as the Estonian specifications referred to below, with more emphasises added on a digital signature's security and trust features.

Based on the above considerations, the task of forensic examination of DSDs is likely to arise in cases involving business and administration, notwithstanding any other criminal, administrative and civil cases, where application of forensics is also well warranted. These cases can, for example, involve e-Discovery of electronically stored information [29], or the examination of mixed business-related records of both paper and digital types. Electronically stored information includes electronic records, which are sometimes backups of documents, but can also be described as data that has been captured and fixed for storage and manipulation in an automated system and that requires the use of the system to render it intelligible by a person [49]. Even though personal certificate bound DSDs cannot be generated in a completely automatic fashion, they are part of the electronically stored documentation retained within the course of business. As circumstances imply, this data can be the subject of forensic examinations, including high-profile cases of fraud, corporate malfeasance and insider trading [23].

Two points are most relevant when the forensic examination of signatures and documents is concerned: identification of forgery and document attribution. Indeed, forensic examinations of documents involve the examination of documentary evidence in order to determine these two properties, i.e. authenticity or authorship [5]. Different definitions of attribution exist, for example defining attribution as a subset of interpretation associated with determining causality, and it is largely about the interpretation of things that lie outside of the digital realm in terms of traces that exist within the digital realm [25]. In this thesis the author is looking simply for any data which is helpful in learning something about the signer of the document and their environment. As regards the question of falsification of DSDs, more precisely by breaking their cryptographic defences, no practical compromises are known [26, 27, 45] and therefore no examination of falsified DSDs is possible.

The questions related to retrieval of attributive data from existent or deleted but recoverable DSD containers remain valid and the question of practical identification of forgeries must be set aside.

3.3 Container Standard ASiC and National Implementation BDOC

The latest ETSI standard 102 918 V1.3.1 (2013-06) on Associated Signature Containers (ASiC) specifies the use of container structures intended for binding together a number of signed objects (e.g. documents, XML structured data, spreadsheets, multimedia contents) into one single digital container based on ZIP and supporting certain types of signatures

[11] The so-called recitals are part of an EU legal act's preamble, providing invaluable background and interpretation and published in the Official Journal.

[14]. These types include the XAdES signature [19], with which Estonian national specifications comply, and which is relevant for this thesis. As explained in the standard, ASIC containers are structurally similar to the OCF (OEBPS Container Format) type of containers, which were originally designed for use in eBooks, but have been adopted as the basis for other containers including that used by ODF (Open Document Format - Open Office) and UCF (Universal Container Format by Adobe Systems) [14]. These observations, gathered from the introductory chapter of the standard, are quite relevant from a forensic standpoint, specifically when file carving is concerned.

An ASIC container has certain internal structures, including a root folder for content and a special 'META-INF' folder for metadata about the content, including associated signatures. The ASICE type of container is the type specifically relevant for this thesis. This container type can hold multiple data objects signed by one or more signature structures and must have an uncompressed 'mimetype' file containing the data identifying the container type, which is 'application/vnd.etsi.asic-e+zip', situated at offset 38. The 'mimetype' file provides the support for 'magic numbers'[12] and is subjected to certain rules [14]. Standard's Annex A.1 [14], while repeating the principles already specified in the document's main body, provides more detailed rules on 'mimetype' implementation, according to which it:

(1) has to be the first file in the archive;

(2) cannot contain 'Extra fields' (i.e. extra field length at offset 28 shall be zero);

(3) cannot be compressed (i.e. compression method at offset 8 shall be zero);

(4) the first 4 octets shall have the hex values: '50 4B 03 04'.

An ASICE container file is assigned an IANA[13] registered MIME type with 'magic numbers' of '0:PK,30:mimetype,38:vnd.etsi.asic-e+zip' [15]. These specified rules are repeated, when appropriate, in later, more specific, standard documentation [16] and therefore must be considered valid up to this point. Immediately it can be observed that the registration contains a syntax error because the third length qualifier must be 50, not 38, alternatively 'application/' must be added. From the point of view of RFC 4288, section 8 [56], which is specifying the procedures for registering media types, the syntax for the ASICE 'magic numbers' is incorrect by having only media subtype listed, without identifying the principal media type. An example of a correct 'magic numbers' representation for a closely related type of file is '0:PK0x030x04,30:mimetype,38:application/epub+zip'[14]. While creating a signature based on ASICE media type registration, the author will appropriately compensate for this syntax error.

Observing actual DSD containers of ASICE type first as handpicked samples and thereafter in a set of a few thousands of naturally occurring documents, we will later learn that they do not necessary follow rules (2) and (3). As a consequence, 'magic numbers' based on an IANA registry entry frequently do not work. Later tests will suggest that IANA 'magic numbers' or file signatures will fail in approximately 62% cases of examined DSD containers. No other 'magic numbers' are published at IANA.

[12] For explanation of the term see IETF RFC 4288: "Media Type Specifications and Registration Procedures", p. 4.11, https://tools.ietf.org/html/rfc4288, retrieved on 15.03.2018.
[13] Internet Assigned Numbers Authority, https://www.iana.org, retrieved on 17.05.2018.
[14] https://www.iana.org/assignments/media-types/application/epub+zip, retrieved on 03.05.2018.

According to the standard, signatures associated with data objects are contained in one or more '*signatures*.xml'[15] XML files in the container's 'META-INF' folder. These signature files contain one or more XAdES signatures. In an XML signature, signed data objects need to be referenced, directly or indirectly, with a set of '<ds:Reference>' elements [14]. These rules make examination of sample files easier because XML elements do include references to their specifications.

Relevant Estonian specifications declare full compliance with the referred ETSI ASiC standard and with the XAdES signature standard [19]. Altogether, the BDOC container and XAdES XML signature follow the standards and protocols listed below [17, 18]:

- ETSI TS 101 903 v1.4.2 – XML Advanced Electronic Signatures (XAdES) and its Baseline Profile ETSI TS 103 171;
- ITU-T Recommendation X.509;
- RFC 3161 – PKIX Time-Stamp protocol;
- RFC 6960 – Online Certificate Status Protocol;
- ETSI TS 102 918 v1.2.1 - Associated Signature Containers (ASiC) and its Baseline Profile ETSI TS 103 174. The latter is in turn based on OpenDocument standard part OpenDocument-v1.2-part3 – Packages.

From the point of view of the container, the Estonian specifications do not foresee any additional or deviating rules, but confirm the rules already set in the ETSI standard, namely that a 'mimetype' file shall be present in an uncompressed form with contents of 'application/vnd.etsi.asic-e+zip'. Earlier Estonian specifications, version 2.1:2013 [20] as well as version 2.0:2013 [21] also confirm that that 'mimetype' is present in an uncompressed form having a value of 'application/vnd.etsi.asic-e+zip'. Both BDOC 2.1 and 2.0 specifications declare adherence to ETSI standard's Annex A.1, which not only means that the contents are not compressed, but mimetype contents must be at a specified offset from the beginning of the file. Later practical examination will demonstrate however, that one of two "official samples"[16] of version 2.1 BDOC files as well as the sample available for version 2.0 have 'mimetype' contents compressed. Interestingly, the documentation of libdigidocpp [22], which is a C++ programming library for handling document digital signing and verification, while specifying the contents of the 'mimetype' file does not set more requirements.

Estonian BDOC specifications concentrate on rules for verification of the signer's certificate at the time of signing, which involve time-stamping or time-marking, depending on the particular solution. Specifications require the XML signature to include an OCSP responder certificate and the signer's CA certificate, as well as the value of the OCSP response. In more specific cases when time-stamping is used, the signature must contain the TSA (Time Stamping Authority) certificate, as well as specifying encodings for those additional time-stamps, which is ASN.1 "der" encoding. Specifications also regulate how XML elements contain the signer's X.509 Certificate [17], which holds crucial attributive data about the signer. Estonian DSD specifications are especially relevant in this work because a practical shortage of ASICE sample files leads to their observation being limited to samples generated by Estonian DSD signing applications, such as DigiDoc3, and a few others collected by

[15] Among the official sample DSDs discussed later BDOC v 1.0 uses 'signature*.xml' naming scheme.
[16] Samples available at https://www.id.ee/?lang=en&id=36161, retrieved on 13.01.2018.

web-scraping. Overall, there is no easy way to independently generate massive amounts of different DSD samples because SK ID Solution[17] provided test ID-cards face limitations[18].

It can be reasonably expected that ASICE sample files generated by popular Estonian applications would stick closely to BDOC specifications with the difference that an ASICE XML signature contains a time-stamp instead of a time-mark as in the case of BDOC[19]. This difference will cause an ASICE XML signature to hold one more encapsulated ASN.1 objects. An ASICE XML signature will include, in addition to an OCSP confirmation of the validity of a signer's certificate, the Time Stamping Service's time-stamp response. BDOC, on the other hand, will have both proof of time of signing as well as proof of validity of the signer's certificate within a single OCSP response, i.e. time-mark [22].

3.4 File Signatures and Carving-Based Recovery

Before extraction of data from a file can take place, the file's data must be located in the storage media either by means of file system records or otherwise. Within the course of DFE it is frequently the case that a file is deleted or hidden and is residing in the parts of the storage media unallocated by the file system. Sometimes the file system itself is gone or damaged. In situations like these the file cannot be found by the file system, even if the file's data is still present on the media. Carving techniques can be applied instead to recover the data [3, 4], with varying success. Carving of files from storage media is therefore an essential aspect of digital forensics. The process of carving is usually defined as recovery of data from "raw" information, as opposed to the recovery of data from the file system metadata [3, 4, 47]. Carving makes use of the file header, which are certain bytes indicating the beginning of recoverable data. Carving can also make use of the file footer, which are certain bytes at the end of the file, or takes some other approach for identifying the end of the recoverable data. Data in between of those two points, i.e. header and footer, is then extracted and saved to a new file, which is the carved file [4, 47]. This approach works best with contiguous clusters, while carving from non-contiguous clusters is by far a more arduous task. Garfinkel [3] performed large-scale analysis of files over a large collection of "classic" hard drives, containing predominantly FAT and NTFS file systems. This 2007 research indicated that only 6% of the files surveyed were fragmented, i.e. non-contiguously stored.

For carving to work, the header of the original file, and, depending on the carving technique used, some identification of the end of file must be obtained. For many types of files, the so-called 'magic numbers' can be transferred into header signatures [47]. 'Magic numbers' are byte sequences that are always present at a given place in the file and thus can be used to identify entities as being of a given media type [56]. It is possible that for certain types of files the 'magic numbers' are provided in specialised sources, including IANA's Media Types Registration[20]. Still, file headers do not depend on the existence of these 'magic numbers' and their signatures can be freely constructed based on examination of the contents of

[17] According to https://www.sk.ee/en/about (retrieved on 07.03.2018), SK ID Solutions is the partner of the Estonian state in issuing certificates for national identity documents (ID-card, Mobile-ID, Digi-ID, residence permit card and e-resident's Digi-ID).

[18] https://www.sk.ee/en/services/testcard, retrieved on 16.03.2018.

[19] See "What's the difference between the digital signature formats .ddoc, .bdoc and .asice?" https://www.id.ee/?lang=en&id=37370, retrieved on 15.03.2018.

[20] The page for Media Type Registration is https://www.iana.org/assignments/media-types/media-types.xhtml, retrieved on 03.05.2018. Note that not all of the file types, for which Media Type is registered have 'magic numbers' descriptions supplemented.

the files [47]. File footers, on the other hand, have no relation to 'magic numbers', but they too can sometimes be discerned by looking at a file's contents.

While contemplating file carving one must take into account that many file systems allocate file data on disk media in units called blocks [47] or clusters. The start of a file or the file header is located at the beginning of a cluster. To speed up the carving process it is suggested that searching for the file header could be made in the first few bytes of a cluster [4], as opposed to reading the entire cluster-size of data. It is also noted that the data does not fill those clusters exactly, leaving slack space, which is the unused space within the last cluster allocated to a file [47]. A source indicates that in the versions of MS Windows in use today this slack space is "empty", in other words, it is filled with hexadecimal '00's [53]. Later in this thesis this will play a role in carving DSDs from NTFS formatted volumes.

Digambar et al [4] describe the workflow of file carving and summarise several carving techniques. File carving typically works by reading into memory a pre-defined portion of the media or media image under examination [47]. Each chunk of data read from the media is searched for the file header. If a matching header is found, then the corresponding file footer is searched for. This method is called "header/footer carving" [4] and is useful in carving many file-types, which, in addition to the file headers, have identifiable file footers. Typically, once the header is located, a file carving tool will search for the footer of the file until one is found, or a file size limit is reached. The data between the start of the header and the end of the footer, or, in the absence of the latter, of the administratively set maximum size, is extracted and a recovered file is created [47]. The method of carving relying on maximum file length is called "header/maximum file size carving" and can be used for carving any file type, especially partially overwritten files [4]. Another, more specialised carving technique is "header/embedded length carving" [4], which relies on reading the length of retrievable data (file size) from the data itself. This way, certain file-types can be carved, for example certain versions of PDF files. Another method of file carving is "carving with validation" [4], which relies on searching for a file type specific validator. This is done in addition to common carving techniques such as identification of the header and the footer, in order to minimise false positives. Examples of files to which this method can be applied include some image file-types as well as PDFs.

It is suggested that certain types of files, including ZIP files, can be found and successfully recovered using yet another carving method, which is "file structure based carving" [4]. In case of ZIP files, this method relies on the existence of certain structures, which are pertinent to ZIP format. These are Local File Headers, which precede the beginning of each file stored inside a ZIP archive, and Central Directory situated at the end of a ZIP file, which consists of records for each archived file. The Central Directory ends with Central Directory End Record, located at the end of ZIP file [1]. These structures are the most relevant to the question of carving signatures and are schematically depicted in the figure below.

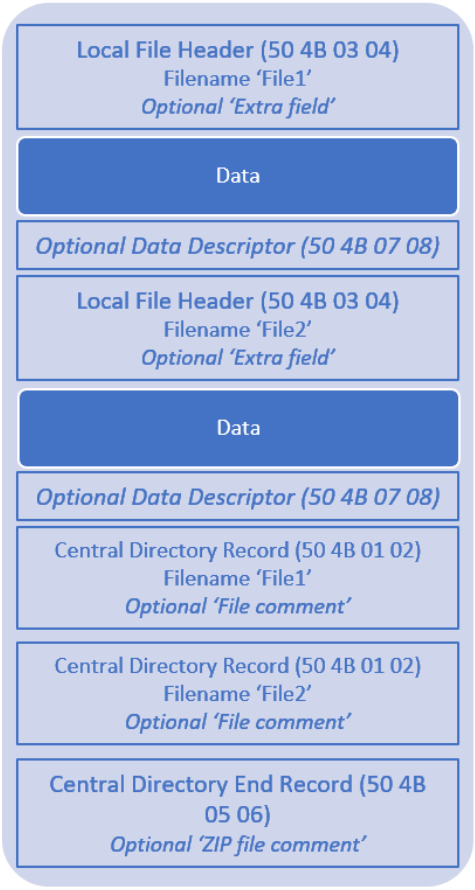

Figure 1. Schematic representation of the structures in a ZIP archive

A more detailed overview of ZIP file structure is provided in Annex 1.

From the point of view of carving ZIP files, it is relevant that a ZIP file starts with Local File Header and ends with Central Directory End Record. Sources [4, 47] suggest that algorithms for carving ZIP files work incrementally by parsing each Local File Header, which have predetermined headers of hexadecimal '50 4B 03 04' (ASCII-hexadecimal representation of 'PK\x03\x04'). In each Local File Header an algorithm is parsing the flag indicating the compressed file size for the stored (archived in ZIP container) file. To arrive at the beginning of the next stored file, an algorithm adds, to the compressed file size, the length of the Local File Header itself, which also includes the name of the stored file as well as the optional 'Extra field', described in more details later in this thesis. Finally, an algorithm arrives at the last structure of the ZIP file, the Central Directory. As stated above, a ZIP file ends with Central Directory End Record, whose beginning is identifiable by hexadecimal '50 4B 05 06' (ASCII-hexadecimal representation of 'PK\x05\x06'). Central Directory End Record may have varying size due to the presence of the optional 'Zip file comment'. To determine the correct length of this end-structure (sometimes called "the trailer" of the file [48]), a carving algorithm reads the value of the flag indicating the length of the comment field in Central Directory End Record [4, 47]. In this way, for example, 10 iterations may be required to parse a sample ZIP archive containing 9 files [47].

At a certain point in this thesis the author will attempt several techniques, described by the sources referred above. The "header/footer" carving technique will be the main method to

test the signatures created for the identification and recovery of DSD containers. This technique will not require iterations inside the ZIP archive. The choice of this particular technique, which is different from the one preferred in the referred sources [4, 47], relies on understanding that possible scripted tools, as well as the mainstream forensic examination suites can sometimes have basic built-in support for header/footer based carving.

To validate whether this approach works the author will first create new or modify existent signatures making use of the headers and the footers identified in the files' contents. Secondly, the author will write scripts applying these signatures to raw data. To achieve successful carving of DSDs the author will have to overcome the main difficulty in carving ZIP containers, which is the absence of a predetermined footer. As fail-safe the author will make use of most of the techniques identified in referred sources [4, 47]. The carving process will be supplemented with "maximum file size" break. The author will also add the optional choice of reading the size of the end of a ZIP file directly from the Central Directory End Record's relevant flag, instead of relying on the footer. To reinforce the results, the author will also add, at an appropriate point in the scripts, a validation check as suggested in the "carving with validation" technique. This check will not make use of the ZIP structures or flags, but of the structures specific to DSDs.

4 Practical Work Contexts

This chapter outlines the questions raised by the author, which is finding a better signature for carving DSD containers as well as looking for their attributive data. Empiric observation by examining different layers of DSDs in collected samples is the method chosen. To validate their findings the author will launch a series of tests, on both a smaller and a larger scale, applying the identified signature found to carve deleted files as well as extracting attributive data based on the proposed algorithm.

4.1 Problem Statement

The broader area that this thesis is trying to address is what are forensically useful properties of DSDs and how to obtain relevant data based on this knowledge. As identified above, two important tasks arise when examining signed documents. These tasks are attribution and detection of forgery.

The question of forgery of DSDs is a complex one. Cryptographic reinforcement of digital signatures, even though not eliminating the possibility of their compromise, makes it extremely difficult to attack them this way. The latest studies [26, 27] indicate that even though there are certain vulnerabilities as well as issues of "ageing" of cryptographic and hashing algorithms, no practical cryptography-related exploits compromising Estonian eID are known. The situation felt worse in late 2017 with the ID-card chip scare [28]. However, according to the RIA[21] blog [45] no actual cases of compromises have been discovered, which remained so as late as in February 2018[22] and is likely to be so at this moment despite the fact that in April 2018 RIA's partner Cybernetica AS succeeded in cryptographic compromise of eID, the details of which remain secret [55]. Based on these considerations, no samples of forged DSDs could be obtained. As this thesis is highly practical and bases itself on smaller or larger scale examinations of samples, the question of falsification must be left out.

As concerns the question of document attribution it is worth remembering that before any conclusions can be drawn, a forensic examiner must perform many intermediate tasks. These tasks include collection, extraction and preservation of evidence [2]. The common workflow of extraction involves recovery of deleted or hidden data, as well as decoding and indexing of full contents. After data holding potential evidence is collected, extracted and indexed, the content is examined, which is assisted by using keyword searches, content recognition or more complicated forms of text, image etc. analysis. Nothing prevents the examiner from extracting pieces of data, which can reveal something useful under investigation, but which were not left intentionally by perpetrators or persons of interest. Those pieces are sometimes called forensic artefacts, a widely used term without a clearly established definition [54]. Digital signing of a document by the holder of private key does create attributive data in DSD, but this data is not an artefact.

In this thesis author will be looking for ways to recover DSDs and collect various relevant data from their contents.

[21] Information System Authority, https://www.ria.ee/en.

[22] Actual cases of practical compromises are yet to be discovered even with regard to aged DDOC documents based on depreciated SHA-1, according to RIA's Markus Kullerkup 09.02.2018 e-mailed answer to author's request for possible samples of compromised BDOC, ASICE and DDOC documents.

4.2 Research Questions

As identified above, with a deeper knowledge of the relevant data properties of DSDs, identification of attributive data can prove useful in DFE. In this work the relevant DSD containers are of BDOC and ASICE types as implemented in Estonia under RIA's supervision. The review of the properties of those containers, stored on classic HDDs (hard disk drives), attempts to address the following questions:

1. How to recover DSD containers by file-carving separately from other similar containers;
2. What is and how to obtain, information useful for attribution of those DSDs.

To answer those questions, the author will undertake a review of available BDOC/ASICE samples. This will involve hexadecimal and plaintext observation of their ZIP encapsulation, including such ZIP format features as Local File Headers and Central Directory Records. This will also involve a review of the XML contents of their signatures as well as ASN.1 encoded data embedded in those signatures. These steps will be referred to as: review of the outlying layer (ZIP container), intermediate layer (XML signature) and inner layer (ASN.1 encoded data including X.509 certificates). When appropriate, the author will also support this review with corresponding documentation.

While answering these questions, the author will concentrate on solving more practically the following tasks:

- Finding header and footer combination(s) useful for identifying the existent files, as well for data carving from unallocated space using classic methods in such a way as to differentiate DSD containers from similar ZIP containers;
- Finding attributive data, that is meaningful for attribution, in all of the layers stated above, as well as exploring how to extract this data and proposing a suitable algorithm.

4.3 Methods

The method for finding headers, footers and attributive data as described in 4.2 will be empiric observation [30]. As direct empiric observation of digital data must be assisted, the author will use appropriate technical tools. These tools will be listed in the corresponding chapter of this thesis.

The author will collect and additionally generate a small set of sample BDOC and ASICE containers and conduct reviews of their outlying layer, intermediate layer and inner layer.

Samples will be obtained from SK ID Solutions published "official samples"[23]. These samples will be further supplemented by DSDs signed, in relevant formats, by the author using his certificate and various current applications for signing documents. Reviewing ZIP, XML and ASN.1 layers with tools capable of reading and interpreting, in human-readable ways, plaintext and binary data, such as hex editors, XML capable browsers and ASN.1 dumpers, will enable insights into what is relevant for research questions including carving based recovery of containers and obtaining possible attributive data.

As a result, the author will deliver a perfected DSD container carving signature, a representation of header and footer. The author will also create an algorithm explaining how to obtain attribution related data.

[23] https://www.id.ee/?lang=en&id=36161, retrieved on 07.03.2018.

4.4 Validation

As one of the stated goals is to find ways to comprehensively recover DSD containers employing file carving techniques, the author will simulate several scenarios. In the first scenario, the author will attempt carving using mainstream data recovery and commercial forensic examination suites[24]. In the second scenario carving-based data recovery will be attempted using a known ZIP signature, a known DSD signature based on 'magic numbers', and the author's footer-header combination, developed as result of empiric observation. Results will be compared.

These tests will be run over unallocated clusters in a forensic image of a typical HDD uncompressed NTFS volume where a mix of the author's chosen sample BDOC/ASICE files will be placed together with ZIP archives and MS Office documents ('.docx', '.xlsx', '.pptx'). Files will be deleted and the NTFS volume quick-formatted, thus forcing clusters to a filesystem-wise unallocated state. To succeed, the author's proposed signature will have to produce better results than typical ZIP signatures, the official DSD signature, as well as mainstream applications and suites. Under these conditions, successful file carving means recovery of DSD containers separately from any other container of similar type, such as ZIP archives and MS Office documents. To enable file carving with custom signatures, the author will write a script in Python capable of detecting different headers and footers, including all of the above.

Thereafter the author will conduct a larger-scale exercise, which will further validate the findings. To achieve this larger scale, the author will web-scrape[25] a large amount of DSD containers from an open source, in this particular case[26], from a government agency's public document registry. The large set of containers obtained in this way will be supplemented by files of structurally similar types. More precisely, MS Office documents, ZIP archives, as well as ODT, ODF documents and EPUB e-books will be added into the mix, this is because the latter are identified in the ETSI standard as especially similar filetypes, structure-wise. The author will thoroughly document the process of web-scraping and the results of file carving exercises. To succeed, the author's signature will have to recover DSD containers, this time on a larger scale, separately from any other container of a similar type.

As the second stated delivery is creating an algorithm for obtaining attributive contents to collect and document relevant data, the author will create a Python script that applies the algorithm. The author then will test this script as well as mainstream commercial forensic examination suites for obtaining relevant contents. To reach this goal certain data from the innermost layer, which was identified as ASN.1 encoded data containing certificates and responses, must be retrieved. For forensic suites, this test will be done by running "keyword searches" for keywords known to exist. Files will be placed in an image. No file deletion will be necessary at this stage. The author's script, on the other hand, can still be tested over deleted files in both small as well as large scale tests. To succeed, the author's script will have to obtain relevant ASN.1 layer data better than mainstream commercial forensic examination suites.

---

[24] EnCase Forensic Licence Agreement defines "suite" as a collection of modules with module meaning a version of the licensed lroduct designed to increase functionality for certain specific tasks or to serve the requirements of a subset of users.

[25] An official-looking definition of this widely used term is hard to come by, however a less official one can be found at https://medium.com/the-andela-way/introduction-to-web-scraping-using-selenium-7ec377a8cf72, retrieved on 22.03.2018.

[26] Based on Web Browser automation using Python Selenium library.

4.5 Practical Considerations

In-depth knowledge of DSD containers enables their differentiation from other similar ZIP-like files. This is a requirement for successful carving-based data recovery over large data sets or if other limitations exist, for example on the types of data to recover.

The relevance of the recovery of different subcategories of ZIP containers can be shown in the following example. Commercial forensic examination suite X-Ways Forensics' File Type Category for "Microsoft Office XML Data Source" is '2', while the Category for ZIP is '3'. The header signature for MS Office 2007+ documents is based on an internal structure specific to MS Office documents, while the signature for ZIP containers is based on classic ZIP header, in ASCII-hexadecimal representation at certain offsets[27]. Later tests will demonstrate that this forensic suite successfully separates MS Office documents and ZIP archives in file carving results.

While the usual sources such as file signature tables, supplied with forensic examination suites, or PhD G. C. Kessler's web-published signature tables [48] do not list specific DSD containers' headers and footers, a source for such a signature is the IANA media type registration. This source was identified above as result of a thorough review of corresponding documentation. Later tests will suggest that most DSD containers do not follow the ETSI ASiC standard-mandated header.

Keyword searches in both contents and metadata, when multi-layered and encoded files are concerned, can be improved by following their internal structure, for example first uncompressing files and thereafter indexing their contents. Knowledge about the structure is thus crucial in forensic examinations because they may sometimes have a large scope [31] in terms of size of data. This knowledge is equally useful where strict limitations are imposed on DFE, which is not an entirely implausible scenario for a criminal search warrant [32] or in an e-Discovery-like scenario [33]. In such circumstances, insufficient understanding of DSDs, if they have significant bearing in the dataset under DFE, will result in an examiner's failure.

DSDs have a radically different purpose and are inherently more complex than traditional office documents of non-proprietary formats. Yet not only are they not supported by forensic examination suites, but they are lacking in sources of forensic information, as opposed to mainstream "office" documents, for which there are forensic overviews [34, 35] available. Discussions held by the author with digital forensic experts of the Estonian Forensic Institute in November 2017 largely confirmed the above considerations.

[27] Signatures and file categories are described in files 'File Type Categories.txt', 'File Type Signatures Search.txt', inside main folder where X-Ways Forensics v 19.5 executable 'xwforensics64.exe" is located. Header entry for MS Office 2007+ is '_Types\]\.xml' at offset 38. Footer is "~14", which means that algorithm no. 14 is used to locate the end of file. Header entry for ZIP is 'PK\x03\x04|PK00|PK\x05\x06' at offset 0. Footer is "~14", which means that algorithm no. 14 is used to locate the end of file.

5 Scope and Limitations

This thesis touches upon the multidisciplinary field of digital signatures. Digital signatures rely on complex tangible and intangible means and concepts involving PKI (public key infrastructure), secure hardware, and law and public administration. This thesis concentrates solely on forensic applications relevant to specific forensic tasks and the conclusions are true only for the types of DSDs examined in the specific environment stated.

5.1 Cryptography

As described above, relevant digital signatures are based on the cryptographic assurance of authenticity, of which non-repudiation is the most important. 117 different attacks on signatures, notwithstanding their practicality in our case, are classified in a 2013 source [43]. Two base scenarios lead to non-repudiation failure: private key compromise and compromise of the signature authentication function. Interestingly, the Digital Signature Act of 2002 accounted for the first type of compromise, while current Electronic Identification and Trust Services for Electronic Transactions Act[28] does it in a softer wording, referring to possibilities of using the private key of the certificate without the consent of the certificate holder in several scenarios. In practice these attacks, i.e. social compromise of eID, must happen almost daily. As this is written on 22nd of March 2018, the latest criminal case published in the Court Registry[29] is from 01st of February 2018 and is describing counts of unauthorised use of another person's ID-card and "passwords" (likely pin codes) to sign lease contracts [44]. The Court Registry lists many similar cases. Despite the apparent frequency of private key compromise and existence of many other potential vectors, successful cases of cryptographic related attacks, for example pre-image attacks, on national signatures are unknown [26, 27, 45]. As samples of compromised or falsified DSDs are not available, no such samples are reviewed in this thesis.

The scope of this thesis does not cover the area of forgery of digitally signed files, or, in a broader sense, attacks directed at their compromise, including any issues related to cryptography. Even though in this work the author decodes, to the extent necessary, data from certificates and responses, which are part of the broader cryptographic framework, this work remains agnostic to the algorithms in question, being only interested in the relevant data stored in the signatures. On similar grounds, packaging algorithms such as ZIP's deflate, encoding schemes such as Base64, and format languages (XML, ASN.1) are not the topic of this thesis.

5.2 Scripts and Third-Party Tools

Some parts of validation of the findings of this thesis include Python scripts for, essentially, extremely simplified mock-ups of forensic examination. Standard Python libraries are favoured. In some cases, external libraries are used, for example Pandas[30], which is used for documentation and aggregation of results. In all cases additional plaintext CSVs are created in parallel.

The scripts written by the author are delivering described results as applied to particular problems and datasets. The author is not legally responsible for the results of applying those scripts to different datasets or problems. Scripts developed by the author for the purposes of

[28] Consolidated version's English translation available at https://www.riigiteataja.ee/en/eli/ee/Riigikogu/act/527102016001/consolide, retrieved 22.03.2018.
[29] https://www.riigiteataja.ee/kohtulahendid/koik_menetlused.html, retrieved on 22.03.2018.
[30] https://pandas.pydata.org, retrieved on 29.03.2018.

validation of this work are intentionally verbose and are not object oriented, unless otherwise apparent. They are not intended to be, computationally, efficiency-optimised.

The author is aware of the MD5 hashing algorithm's cryptographic weakness [26]. In all cases MD5 is used as a unique identifier for files or data where the expected risk of collision is considered negligible.

Testing done with third party digital forensic solutions assumes a standard workflow intended for similar forensic examinations with these tools. This does not mean that the same tools will not perform better using different workflows or in combinations with other tools. The choice of the mainstream commercial tools is based on the same tools chosen for testing in a referred source [42].

5.3 Signature and Attributive Data

The proposed signature has been tested under Windows 10 Home 64 for file carving in an NTFS v 3.1 formatted volume of a hard drive disk. The application of the header signature is dependent on the drive's geometry. The footer signature has been successfully tested in zero-slack media as described in the corresponding chapter. Only extraction of data described as attributive has been tested.

5.4 Anti-Forensics Techniques

Suspects and persons of interest in a DFE can impede an investigation by destroying or modifying evidence. This practice is sometimes called anti-forensics. Gül et al [57] provides an overview of some anti-forensic techniques including those which are directed at manipulating the file header, which is sometimes referred to as "transmogrification". This technique could seriously impede a DFE relying on header based signatures for identification of files. The headers could be manipulated by applying a specialised tool, or by manually changing the hexes, causing a forensic examination suite to incorrectly detect file type.

In this thesis the author proposes a signature for detecting DSD files. Though this signature is specific to a particular type of file and will not detect generic ZIP archives, it does rely on a typical ZIP header of ASCII-hexadecimal representation 'PK\x03\x04'. Assuming that the use of anti-forensics is to obstruct an investigation while still retaining the data, the most straightforward and the easiest way of hiding a DSD file from being detected by the automated tools is changing its ZIP header, effectively falsifying the file type. For example, changing the file type effectively into a Microsoft CAB file by replacing the current header with 'MSCF' [48] could, in the author's opinion, make the file look inconspicuous. This would also render a custom DSD signature powerless. Despite that and assuming that transmogrification does not go deeper, the file can still be identified by the existence of DSD specific contents, such as the XML signature file 'META-INF/signature*.xml'. To achieve that, the signature proposed later in this thesis would have to be modified starting at offset 4 and loosing the 4-hexes long ZIP header of 'PK\x03\x04'.

Changing the contents of the signed files or of the XML signature itself, on the other hand, is easily discoverable as warning messages about signature invalidation are displayed in DSD signing applications.

5.5 Legal Considerations

With regard to web-scraping of a government agency's public document registry looking for publicly available documents, the author is aware that the 'robots.txt' of the particular domain does not allow bots. The complexity of the legal issues associated with this situation exceeds the scope of this work. For this thesis the author has commissioned legal analysis

from a national legal bureau highly experienced and competent in the field of data protection law. In this analysis it was found that this particular thesis does not infringe, in any way, data protection rules and regulations. The legal analysis is not attached to this work but is made available at the defense. To protect personal data included in the scraped documents, any such data is left out of this publication, except for highly aggregated impersonalised summaries. The scripts used for web-scraping are not made public.

6 Examination of Sample Containers

This chapter describes collecting and reviewing contents of both official DSD samples, which are the files made available at the certification authority's Web page, and samples of the author's own documents signed in several different applications. The structure of a ZIP container as well as an XML formatted signature and ASN.1 encoded objects inside signature files are reviewed with the purposes of this thesis in mind, which are finding a better carving signature and learning about forensically attributive data.

6.1 Sample Set

SK provides a sample of supported DSD containers for all major versions. These containers are currently listed and described on an SK ID Solutions dedicated Web page titled "*DigiDoc file container format support cycle (life cycle)*". The files in question and their container versions are 'BDOC-1.0.bdoc' v 1.0, 'BDOC2.0.bdoc' v 2.0, 'BDOC2.1.bdoc' v 2.1 (TM) and 'BDOC2.1_TS.asice' v 2.1 (TS). MD5 hashes for these and for the following files are provided in a separate Annex IV to this thesis.

These samples are supplemented by the author's signed DSDs and are described in Table 2 below.

As for the document that was to be signed, the author used the Estonian Chamber of Commerce and Industry's example document titled "*Power of Attorney*"[31]. This file is named 'Volitus_firma_esindamiseks_inglise_keeles.pdf' and its contents filled out with fictitious information, which can be summarized as "Ivan Orav" from the company "Vanad Tuttavad OÜ" representing a fictitious foreign company "Market-Research Oy". The unsigned document is converted to a PDF prior to signing.

If the application used for signing allowed for additional fields, the following information was entered by the author at the signing process.

Table 1. Additional fields in the author's signed DSD containers

Field	Value entered
Role	Signator
Resolution	Agree
City	Tallinn
State	Harjumaa
Country	Estonia
Zip	13628

The signed documents as well as the environment used for signing are described in the table below.

[31] https://www.koda.ee/et/tooriistad/valiskaubandusalaste-dokumentide-naidised, retrieved on 10.02.2018.

Table 2. DSDs signed by the author

Application	Application Version	OS	Signed with	DSD name with extension
DigiDoc3	3.13.4.1515	MS Windows 10	Id-Card	DCC.bdoc
DigiDoc3	3.13.4.1515	MS Windows 10	Mobile Id	DCM.bdoc
DigiDoc3	3.13.4.1515	MS Windows 10	Id-Card	DCC.asice
DigiDoc3	3.13.4.1515	MS Windows 10	Mobile Id	DCM.asice
app.digidoc.ee	n/a	n/a	Id-Card	DAC.bdoc
app.digidoc.ee	n/a	n/a	Mobile Id	DAM.bdoc
app.digidoc.ee	n/a	n/a	Id-Card	DAC.asice
app.digidoc.ee	n/a	n/a	Mobile Id	DAM.asice
DigiDoc Finestmedia	2.0.14	Android 6.0	Mobile Id	DMM.bdoc
DigiDoc3	3.13.4.1515	MS Windows 10	Empty	DCE.bdoc
DigiDoc3	3.13.4.1515	MS Windows 10	Empty	DCE.asice

The applications and operating systems used to sign the documents listed in Table 2 are in more details described below.

(1) DigiDoc3 application: qdigidocclient version 3.13.4.1515, released 14.11.2017, base version: 17.11.0.1762, under MS Windows 10 Home 10.0.16299. DSDs in this client application were signed using a Gemalto ID-card reader (s/n I17C01474009795) as well as Mobile ID under Android 6.0 Kernel v 3.4.0.

(2) Web based application of https://app.digidoc.ee, domain records indicate 'UAB ESTINA'. DSDs in this client application were signed using a Gemalto ID-card reader (s/n I17C01474009795) as well as Mobile ID under Android 6.0 Kernel v 3.4.0.

(3) DigiDoc v 2.0.14 by Finestmedia, signing with Mobile ID under Android 6.0 Kernel v 3.4.0. The application has the Mobile ID signing document's format set to BDOC and itself generates the filename of the container, based on the name of the file to be signed. In this particular case, the application also added a string "1518515471170" to the end of the filename. For the purpose of this examination the filename was changed to the one listed in Table 2.

After the creation of the set of DSD samples, their contents can be observed based on the layers identified above. Figure 2 provides a schematic[32] depiction of the observational stages of reviewing of the sample containers' contents.

[32] Note that in the figure, certain elements such as the signed documents themselves (DOCX, PDF etc), as well as some of the embedded certificates are not shown.

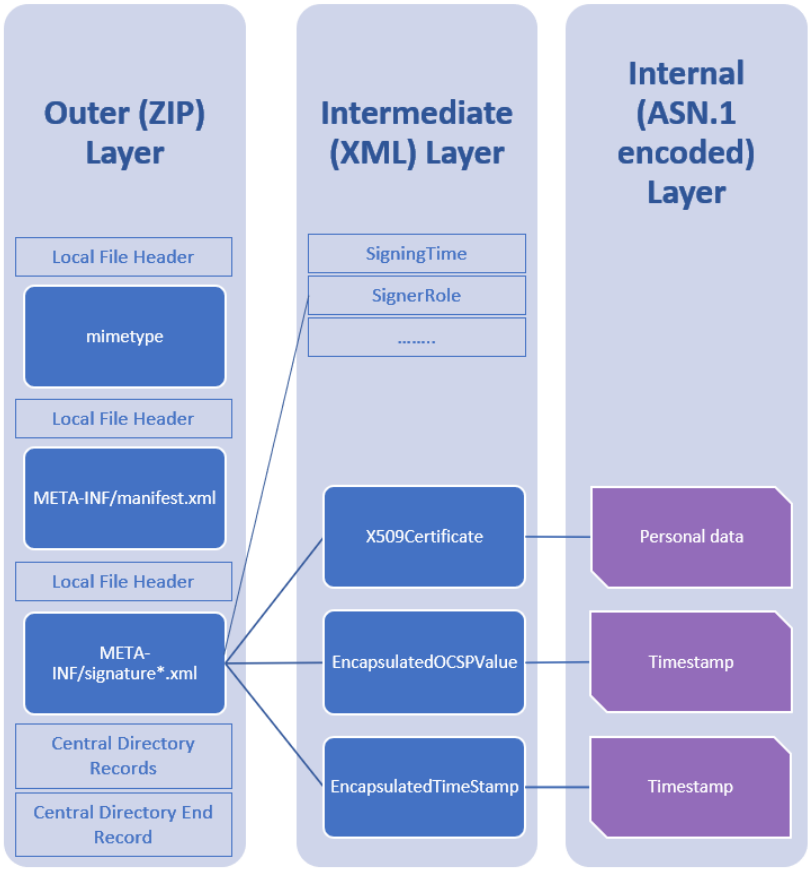

Figure 2. Stages of decoding and reviewing of the contents of the sample DSDs represented as layers

6.2 Outer (ZIP) Layer

ZIP can be described as a file format, universally used to aggregate, compress, and encrypt files into a single interoperable container [1, sec. 1.2.1]. According to ZIP specifications [1] ZIP containers start with a Local File Header, which begins with hex '50 4B 03 04' and is describing a single file inside the container [1, sec. 4.3; 36]. The Local File Header also specifies whether the file is zipped or not. After the Local File Header, which contains a plain-text filename, the file's contents follow. ZIP containers end with Central Directory Records, the start of which is designated with hex '50 4B 01 02' carrying additional information regarding file attributes. Both the Local File Header as well as the Central Directory Record may include 'Extra field', the Central Directory Record may also include 'File comment'. The Central Directory ends with Central Directory End Record, starting with hex '50 4B 05 06', which may also have an optional 'Zip file comment' field. If no optional fields are used, Local File Header, not including filename, is 30 bytes long and Central Directory End Record is 22 bytes long.

As discussed above, DSD containers must have a 'mimetype' file, which comes as the first file inside the archive. Standards and specifications for at least ASICE, BDOC 2.0 and BDOC 2.1 mandate that 'mimetype' contents are not deflated. Some of the sample containers ('BDOC-1.0.bdoc', 'BDOC-2.0.bdoc', 'BDOC2.1.bdoc') were observed to have the Local File Header flag at offset 9 set to '08' indicating that the file 'mimetype' is zipped. In other files the flag was set to hex '00' indicating that file 'mimetype' is not zipped [1, 36]. The sample file 'BDOC21-TS.asice' has 'mimetype' contents undeflated, but because of a, seemingly unmandated, use of 'Extra field', has the start of contents at offset 66 instead of

the standard's 38. Out of the official samples, no files were fully compliant with the part of the standards and specifications regarding the first Local File Header, which also suggests that they could not be located using the official 'magic numbers' described above.

As previously noted, three containers have 'mimetype' file contents zipped with the first part of the zipped contents matching for all three files. This is relevant from the point of view of creating a carving signature for identified headers.

Table 3. Certain containers 'mimetype' file zipped contents

Container	**Zipped 'mimetype' contents in hex view**	
	First part (matching)	Second part (different)
BDOC-1.0.bdoc	4B 2C 28 C8 C9 4C 4E 2C C9 CC CF D3 2F CB 4B D1 4B	4A C9 4F D6 35 D4 33 00 00
BDOC-2.0.bdoc	4B 2C 28 C8 C9 4C 4E 2C C9 CC CF D3 2F CB 4B D1 4B	2D 29 CE D4 4B 2C CE 4C D6 4D D5 AE CA 2C 00 00
BDOC2.1.bdoc	4B 2C 28 C8 C9 4C 4E 2C C9 CC CF D3 2F CB 4B D1 4B	2D 29 CE D4 4B 2C CE 4C D6 4D D5 AE CA 2C 00 00

The other sample files, which include containers generated by the author, appear to have 'mimetype' files' contents not zipped and, according to specifications, situated at offset 38 from the beginning of the container, with one exception where the contents were at offset 66.

Some of the files ('DMM.bdoc', 'DAC.bdoc', 'DAM.bdoc', 'DAC.asice', 'DAM.asice', 'BDOC-2.0.bdoc', 'BDOC2.1.bdoc') make use of Data Descriptors, which are optional and are situated after the end of file data. Data descriptors have their own headers of hex '50 4B 07 08'. These structures appear to hold flags related to validations of the stored file data and its size [1].

Several of the sample containers make use of the Central Directory Record's 'File comment', or the Central Directory End Record's 'ZIP file comment' to describe system related metadata. For example, Central Directory's End Record for 'DAC.bdoc', which was signed by the author using the Web application of app.digidoc.ee contains the string '*LIB DigiDoc4j/DEV format: application/vnd.etsi.asic-e+zip signatureProfile: ASiC_E_BASELINE_LT Java: 1.8.0_111/Oracle Corporation OS: Linux/amd64/3.10.0-514.el7.x86_64 JVM: OpenJDK 64-Bit Server VM/Oracle Corporation/25.111-b15*'. This might suggest that this application is running on an operating system with CentOS33 Kernel 3.10.0-514.el7.x86_64. Such a data entry may, in the author's view, be, forensics-wise, considered an "artefact", as it provides additional data about the signer of the document or, in this particular case, about the signer's environment. Notwithstanding classifying this data as an "artefact" or as more general attributive data, it might prove useful in DFE. In fact, OS and application (Web Browser) related information has been used by prosecutors as evidence to attribute an attack to a suspect [38]. Programming documentation [37] of DSD suggests that inclusion of this system related metadata is intentional for debugging purposes.

As regards examined samples' footers, they end with Central Directory End Record's standard ZIP header of hex '50 4B 05 06' with some of the samples having the 'ZIP file comment'

[33] Look at kernels listed at https://buildlogs.centos.org/c7.1611.01/kernel/20161117160457/3.10.0-514.el7.x86_64, retrieved on 08.03.2018.

field filled. These sample files apparently end with alphanumeric [58] data, while containers not using 'ZIP file comment' field end with hexes of '00 00 00 00'.

A more detailed hex-editor review of a sample file 'DMM.bdoc' as well as a review of Local File Headers' 'Extra field', which is found in 'BDOC21-TS.asice' is provided in Annex I.

HxD Hexeditor 1.7.7.0 and FTK Imager 3.0.1 were used for observation of contents.

6.3 Intermediate (XML) Layer

XML or "eXtensible Markup Language" is defined as a software- and hardware-independent tool for storing and transporting data. The tags and the structure of an XML document are defined by the author, i.e. by the person who created the document[34], or by the applicable specifications.

Reviewing the sample containers' XML contents is made easier by the inclusion in their XML elements of references to their corresponding specifications. References include XML schemas and rules, as well as syntax specific to signature processing. When observing a sample file's signature part, we can use this information to more accurately ascribe its contents. For example, reading signature contents together with specification, we ascertain that XML element '<ds:X509Data>' inside a sample signature contains one or more identifiers of keys or X509 certificates. If thereafter we encounter a '<ds:X509Certificate>' element, we understand that this a Base64-encoded X509v3 certificate [39] with Base64 defined as Content-Transfer-Encoding designed to represent arbitrary sequences of octets in a form that need not be humanly readable[35].

The XAdES or XML based digital signature's standard [19] further expands understanding of XML elements by including those specific to a particular signature format. In sample files, for example, '<xades:SigningCertificate>' contains, among other data, the signer's certificate number and '<xades:SigningTime>' contains the time at which the signer claims to have performed the signing process. The '<xades:SignerRole>' tag is present in some sample signatures, but absent in others. Those signatures where the author entered, during the signing process, optional information, have tags related to 'City', 'StateOrProvince', 'PostalCode', 'CountryName', 'ClaimedRole' filled with the relevant information. It can be well argued that elements such as the role assumed by signer, or time of signing, can have attributive value as exculpatory or inculpatory evidence, similar to what can be observed in some court rulings [38]. Observation reveals that the XML part of the signature does not have a plain-text readable name and personal code of the person who signed the document.

Most other XML signature elements appear, to a large extent, to be describing signing process and ensuring the validity of the signature. For example, the tag '<ds:SignatureMethod>' indicates the algorithm used to sign data in the 'SignedInfo' element. The 'Reference' element points to the data object that was signed, i.e. the signed document. '<ds:DigestMethod>' indicates the digest algorithm (ECDSA-sha256 in the case of 'DMM.bdoc', which was signed by a mobile application) and '<ds:DigestValue>' contains the Base64 encoded digest value. Encoded data embedded in the XML signature of 'DMM.bdoc' are the signer's 'X509Certificate', Certificate Authority's 'EncapsulatedX509Certificate', OCSP responder's certificate and OCSP ASN.1 encoded response. From the point of view of looking for attributive data, XML elements describing the signing process and relevant

[34] Introduction to XML, W3Schools.com. https://www.w3schools.com/xml/xml_whatis.asp, retrieved on 23.04.2018.
[35] Definition from RFC 1512, 5.2. Base64 Content-Transfer-Encoding. https://www.ietf.org/rfc/rfc1521.txt, retrieved on 23.04.2018.

to a signature's validity may have a lesser value. ASN.1 encoded elements, on the other hand, especially the signer's certificate holding the signer's name and personal code as well as time responses, have great value.

As a result of review, the following XML elements are considered the most valuable: 'SigningTime', 'X509IssuerName', 'X509SerialNumber', 'SignatureProductionPlace', 'City', 'StateOrProvince', 'PostalCode', 'CountryName', 'SignerRole', 'ClaimedRoles', 'ClaimedRole', 'ByName', 'ProducedAt'.

Internet Explorer v 11 was used to view XML signatures. Relevant elements are depicted in Annex II.

6.4 Internal (ASN.1 Encoded) Layer

ASN.1 or "Abstract Syntax Notation number One" is defined as a formal notation used for describing data transmitted by telecommunications protocols, regardless of language implementation and physical representation of these data, whatever the application, whether complex or very simple[36].

As noted above DSD-wise this XML signature holds a number of ASN.1 encoded objects whose XML elements include 'X509Certificate', 'EncapsulatedX509Certificate', 'EncapsulatedOCSPValue' and, when relevant, 'EncapsulatedTimeStamp'.

According to the explanations included in the standards, ASN.1 encodes quite complex data types, enabling carrying their messages without concern for their binary representation. ASN.1 encodes this data with a number of different algorithms such as Basic Encoding Rules (BER), Packed Encoding Rules (PER), and XML Encoding Rules (XER) [46]. ASN.1 includes OID object identifiers. OID is defined in an earlier standard as a globally unique value associated with an object to unambiguously identify it [40]. In a later standard, OID is defined as an ordered list of primary integer values from the root of the international object identifier tree to a node, which unambiguously identifies that node [46]. In the sample files' X.509 embedded certificate, as decoded by the Certutil utility, there are several highly relevant OIDs, such as '2.5.4.5', holding the value of the author's personal code.

The embedded 'X509Certificate', which is the first ASN.1 object in an XML signature, is the so-called end entity certificate, defined in the standard as issued to subjects that are not authorized to issue certificates [41]. A practically useful example of an end entity certificate is provided on page 140 of RFC5280[37]. In this example, the user's certificate is signed with 'sha1withRSAEncryption' algorithm, as can be seen from the OID of '1.2.840.113549.1.1.5'. To compare, the author's certificate embedded in the signature of 'DMM.bdoc' signed with 'sha256RSA' has an OID of '1.2.840.113549.1.1.11'. Most relevant to the question of attribution posed in this thesis are OIDs like '2.5.4.3' 'Common name' in the example and 'Sur Name' '2.5.4.4', 'Given Name' '2.5.4.42' and, most likely, 'Serial Number' '2.5.4.5' in the author's signature certificate.

Embedded Certificate Authority and Responder's certificates do not, in the author's opinion have direct attributive information. However, 'EncapsulatedOCSPValue' and 'EncapsulatedTimeStamp' contain useful information concerning the verified time of response, which may be relevant.

[36] Introduction to ASN.1, ITU.int. https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx, retrieved on 23.04.2018.
[37] https://tools.ietf.org/html/rfc5280, retrieved on 17.05.2018.

As a result of review, 'UTF8String', 'PrintableString', and 'UTCTime' datatype objects, which hold values of all of the identified OIDs, in 'X509Certificate' are considered the most valuable data, as well as time-responses in 'EncapsulatedOCSPValue' and 'EncapsulatedTimeStamp', which can be read directly by decoding Base64.

ASN.1 encoded objects were reviewed using MS Certutil v 10.0.16299.15 and ASN1Editor v 1.3.10 and the observed results are described in more detail in Annex III.

6.5 Findings

An examination of sample containers contents provides the opportunity to suggest perfected header signature and improved footer signature, useful in file carving as well as for identifying attributive data and charting out an algorithm for its extraction.

6.5.1 Header and Footer

Based on the above examination, the following container header scheme is devised.

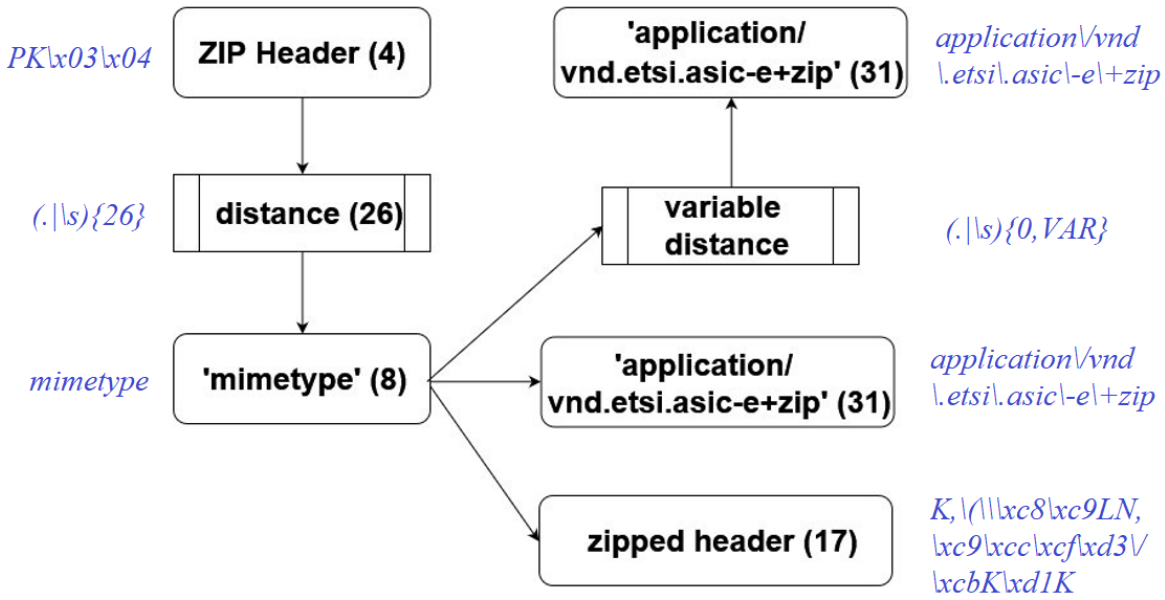

Figure 3. Header scheme

Figure 3 depicts the elements of the header and their corresponding lengths in brackets. ASCII-hexadecimal representation of corresponding signature's regular expression in Pythonic syntax is provided for each element in blue Italic.

For the sample files, the value used for the variable distance 'VAR' was 28. Note that the element "zipped header (17)" identifies the compressed, or "zipped" part of the header's contents, 17 hexes long, identified in Table 3 above.

In this scheme, first, the standard ZIP header of hex '50 4B 03 04' with a length of 4 indicates the beginning of the file, after a distance of 26 there follows filename 'mimetype' ('6D 69 6D 65 74 79 70 65' in hex) with a length of 8. After the 'mimetype' filename entry one of three options takes place:

a) Uncompressed contents 'application/vnd.etsi.asic-e+zip'. Hex representation is '61 70 70 6C 69 63 61 74 69 6F 6E 2F 76 6E 64 2E 65 74 73 69 2E 61 73 69 63 2D 65 2B 7A 69 70', length 31. Such a header would correspond well to the standard man-

dated IANA registered ‘magic numbers’. Note that in IANA registered ‘magic numbers’ an obvious error was made, since the standard requires the word ‘application’ and a slash preceding ‘vnd.etsi.asic-e+zip’, which would also reflect the subjective hierarchy of media types as provided in the registration entry. While writing regex syntax for the header following this official ‘magic numbers’ the author compensated this error by adding extra length (see header signature below).

b) Zipped part of ‘mimetype’ file as observed in ‘BDOC-1.0.bdoc’, ‘BDOC2.0.bdoc’, ‘BDOC2.1.bdoc’. The first several hexes of these files have the same value, which is ‘4B 2C 28 C8 C9 4C 4E 2C C9 CC CF D3 2F CB 4B D1 4B’ with a length of 17.
c) At an unspecified distance, which depends on the length of ‘Extra field’ in Local File Header, unzipped contents ‘application/vnd.etsi.asic-e+zip’ follows. Hex representation is the same as in option “a”, length 31. Observed length of ‘Extra field’ in the particular sample set is 28.

The maximum length of the header is 55 bytes for cases “a” and “b”. For case “c” distance can be measured, but for other similar cases remains undetermined as it was only seen in ‘BDOC21-TS.asice’ and because ‘Extra field’ is allowed to be of variable length by specification.

Based on the above scheme, the following Pythonic (tested with v 3.5) regular expression enhanced ASCII-hexadecimal representation of header signature is proposed:

‘PK\x03\x04(.|\s){26}mimetype(.|\s){0,28}(application\/vnd\.etsi\.asic\-e\+zip|K,\(\\\xc8\xc9LN,\xc9\xcc\xcf\xd3\/\xcbK\xd1K)’

In the signature’s regular expression above note that the distance qualifier from the filename ‘mimetype’ until file contents is set to ‘{0,28}’. This is only true for sample files, however. Otherwise this distance should be set as a variable, which may be larger.

To compare, the Pythonic representation of IANA registered ‘magic numbers’ with the previously identified syntax error compensated by the additional distance ‘{12}’ is:

‘PK\x03\x04(.|\s){26}mimetype(.|\s){12}vnd\.etsi\.asic\-e\+zip’

As regards the file footer, some of the observed sample containers have Central Directory End Record supplemented with ‘Zip file comment’. Because of the comment, the length of the End Record is 249 bytes for those samples, instead of 22 bytes for the samples with no comment. Central Directory End Records with comments enabled end with alphanumeric characters while non-commented Central Directory End Records end in hex ‘00 00 00 00’ for reviewed samples. Based on the referred ZIP specifications, the last two hexes flag the size of the comment; therefore, it is assumed that for those containers hex ‘00 00’ is present at offset 20. On those considerations and taking into account End Record’s local header of hex ‘50 4B 05 06’, the footer follows the logic in the picture below.

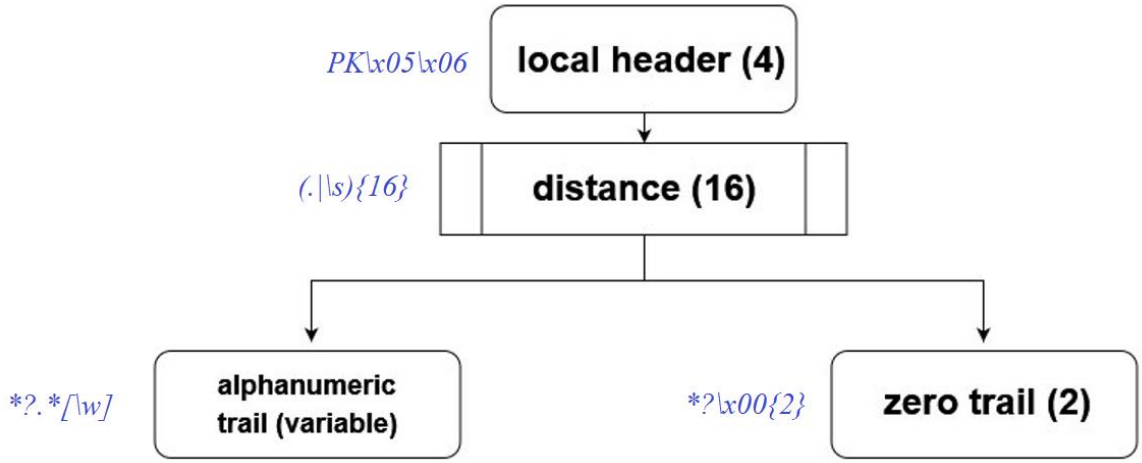

Figure 4. Footer scheme

Figure 4 depicts the elements of the footer and their corresponding lengths in brackets. ASCII-hexadecimal representation of corresponding signature's regular expression in Pythonic syntax is provided for each element in blue Italic.

The Pythonic syntax that would catch the end of the container based on the scheme above:

'PK\x05\x06(.|\s){16}.*?(\x00{2}|.*[\w])'

This footer signature would replace traditional 'PK\x05\x06' + distance 18. Note the non-greedy repetition qualifier '*?'.

6.5.2 Attributive Data

As demonstrated by a review of the containers' ZIP Layer, Central Directory Records may have their optional 'File comment' field filled with information about the system that produced a particular DSD. Equally, such information may be expected in Central Directory End Record's 'ZIP file comment'. Additional information may also be present in Local Header's 'Extra Fields'. This data is considered attributive, which, by definition is data helpful in learning more about the signer or their environment.

Observation of plaintext-readable elements in the XML layer suggest 'SigningTime' XML element as attributive. Elements of interest may also include 'IssuerSerial', 'X509IssuerName' and 'X509SerialNumber'. In the cases where optional information is present as entered in the course of signing, additional attributive elements found are 'City', 'StateOrProvince', 'PostalCode', 'CountryName', 'ClaimedRole', 'ClaimedRole', 'ByName' and 'ProducedAt'. These fields are filled in by the person signing and are not verified. The XML part of the signature does not have a plaintext readable name and personal code of the person who signed the document. The embedded X.509 certificate, however, does.

'X509Certificate' or signer's certificate as observed in the ASN.1 encoded layer contains valuable attributive information, more precisely the name and the personal code of the signer stored as 'UTF8String', 'PrintableString' datatypes with OIDs 2.5.4.3, 2.5.4.4, 2.5.4.42, 2.5.4.5. 'UTCTime' datatype holds the time-period of the signing certificate validity. Additionally, ASN.1 encoded responses in 'EncapsulatedTimeStamp' (ASICE) and 'EncapsulatedOCSPValue' (BDOC) provide timestamps, which, despite being some seconds later than the actual signing, can arguably be considered more trustworthy than XML's 'SigningTime' as they are asserted by a third party. For example, in 'DMM.bdoc' XML 'Signing-

Time' is '2018-02-13T09:54:59Z' (Z stands for UTC + 0), while time recorded in 'EncapsulatedOCSPValue' is '13.02.2018 11:55:09' as shown by Certutil. This data is stored in ASN.1 as 'Generalizedtime' datatype.

Based on the above, the following algorithm for obtaining this data within the course of DFE can be proposed:

Step 1. Read data from file-object or recover container using a carving signature when applicable.
Step 2. Unzip and retrieve signature XML file(s), extract available Extra field of interest from Local File Header, extract ZIP comments from Central Directory Record and Central Directory End Record.
Step 3. Parse XML for data of interest for 'SigningTime', 'IssuerSerial', 'X509IssuerName' and 'X509SerialNumber' as well as user-inserted 'SignatureProductionPlace', 'City', 'StateOrProvince', 'PostalCode', 'CountryName', 'SignerRole', 'ClaimedRoles', 'ClaimedRole', 'ByName', 'ProducedAt'; retrieve ASN.1 objects.
Step 4. Decode Base64 and parse ASN.1 'X509Certificate' for all values of 'UTF8String', 'PrintableString', 'UTCTime' datatypes. Decode Base64 and retrieve time-related responses from 'EncapsulatedTimeStamp' (if present) and 'EncapsulatedOCSPValue'.
Go to Step 1.

7 Validation

In this chapter, the author creates images containing deleted sample files supplemented with similar ZIP archives and MS Office documents, placed in quick-formatted NTFS volumes. The author tests pin-pointed file carving recovery of the deleted DSDs applying a number of forensic and non-forensic data recovery tools, as well as the author's own carving signature, supported by the author's file carving script. The author also tests extracting attributive data, this time from intact files, applying the same mainstream tools and, at a later stage, the author's algorithm for data extraction. Finally, the author web-scrapes a large number of DSDs and tests signature and the algorithm again, to "surgically" file-carve and extract data from a massive amount of deleted real-life containers, mixed together with very similar documents of other types.

7.1 Samples-Based Testing

In the previous chapter an examination of 9 samples signed by the author, 2 empty containers and 4 official sample DSD containers from SK ID Solutions was undertaken. To validate the results the author added 3 DOCX, 3 XLSX 3 PPTX and 3 random ZIP samples downloaded from the Internet. The author also added to the set the unsigned 'Volitus_firma_esindamiseks_inglise_keeles.pdf' (see above) converted to DOCX, and another one zipped as ZIP. In total there were 29 sample files.

The files were placed on an NTFS formatted, uncompressed volume of a size of 2048000 sectors (512 bytes per sector, 8 sectors per cluster) on a regular hard drive (Samsung Barracuda SATA model ST500DM005 s/n S20BJAOCC22298). Raw images as well as E01 (EnCase evidence format) images were acquired from the sectors of the drive containing the volume. The files were thereafter deleted using "shift + delete" while the volume was mounted in Windows 10 Home 64. The volume was quick-formatted, a new volume created of the same size, and a second set of images was acquired.

EnCase Forensic v 8.03.01 was used to acquire both sets of E01 format images of the sectors 0 – 2050047, which is from the 0 sector of the drive to the end of volume, with EnCase FastBloc SE acting as write-blocker to protect the media from accidental change. FTK Imager v 3.0.1 was applied to re-acquire the same sectors 0 – 2050047 images in DD format.

For more details on files, drive and images, forensic examination suites and results of validation, see Annex IV.

Finally, a set of popular data recovery tools as well as mainstream data processing capable [42] commercial forensic examination tools was selected for the file carving as well as for keyword search testing. These software tools are the property and/or trademarks of their respective owners.

Table 4. Tools

Tool	Version	Role	Obtained as
Disk Drill	2.0.0	Recovery	Free version
Recuva	1.53	Recovery	Free version
MS Windows 10 Home	10.0.16299	Keyword search	Licence
EnCase Forensic	8.03.01	Recovery, keyword search	Licence dongle
Nuix	7.4.2	Recovery, keyword search	Trial dongle
X-Ways Forensics	19.5	Recovery, keyword search	Licence dongle
Access Data Forensic Toolkit	6.4.0.70	Recovery, keyword search	Trial licence

These tools recovered deleted files from the quick-formatted volume as depicted in the following table. The success of file recovery was determined by successful unzipping of file - candidates, which was tested with an unzipping script for the large result sets. The results of carving based on an MD5 test are provided in a separate Annex IV.

Table 5. Recovered files by types based on successful unzipping

Type	**Disk Drill**	**Recuva**	**EnCase**	**Nuix**	**X-Ways**	**FTK**	**Files originally**
DOCX	4	4	2	4	4	4	4
XLSX	3	3	0	3	3	3	3
PPTX	3	3	0	3	3	4	3
ZIP + BDOC, ASICE	5 + 0	18 + 0	3 + 0	18 + 0	18 + 0	9 + 0	4 + 15
Other	13 (JAR)		65				
Files total	28	28	70[38]	28[39]	28	19[40]	29

In all cases if a DSD container was recovered, it was assigned an extension by a tool as ZIP type, except in case of Disk Drill, which assigned the extension '.jar'[41] to DSD containers. One ZIP file out of three random samples downloaded by the author from the Internet was an NTFS resident file. As such, it was not possible to recover this file based on the header/footer. Therefore, in this scenario a maximum recovery of 28 files was theoretically possible.

The author then applied the proposed header/footer signatures by creating a Python (v 3.5.3) script (1)[42] taking use of standard 'os' and 're' program libraries, capable of reading file-like objects, including raw (DD) images and "live" drives. This script was structured to recover the starting sectors and end sectors occupied by data. Applied to a file-like object and following standard hard-drive storage media geometry and contiguous data, the script produces two lists, one of starting sectors and another of end sectors, designating the start and the end of candidate files.

While carving data based on sectors can be considered sufficient, more efforts are required to achieve complete matches of carved files to the originals. Algorithms published in various sources [4, 47] are carving ZIP files with increased precision by jumping between local files

[38] Out of 1862 files in total carved by EnCase, 70 files, which passed the unzipping test, were considered candidates. Of those files EnCase assigned extensions '.docx' and '.xlsx' to 5 files. The reason why EnCase recovered so many files seems to be in EnCase's use of apparently every Local File Header inside the ZIP archive as the proper file header. The resulting files are, for the candidates which passed the unzip test, functional ZIP files, but many of them are only partial compared to the originals.

[39] Out of 545 files in total carved by Nuix, 28 files, which passed the unzipping test, were considered candidates.

[40] Out of 269 files and 7 folders carved by FTK, 19 files, which passed the unzipping test, were considered candidates. All these files were assigned '.zip' extension, however they were also categorised differently as documents, spreadsheets, presentations and archives.

[41] Review of a sample JAR file reveals that, similar to DSD containers, the sample file has a 'META-INF' folder containing the file 'MANIFEST.MF', while DSDs have the same named folder containing the file 'manifest.xml'.

[42] https://github.com/raul-nugis/bdoc_carver, 18.03.2018.

in ZIP archives. These algorithms rely on movements between Local File Headers inside ZIP file. At the end, those algorithms analyse the length of the 'ZIP file comment' field, if present. It is noted that algorithms like these are to a great extent more efficient than the one used by a tool called Foremost, which is relying on the unzipping of each included individual file [47]. The author's approach relies, on the other hand, on improving the signature representing the footer to retrieve the end of the whole archive, including the critical notion observed in samples that if 'ZIP file comment' exists, it consists of alphanumeric data. In other cases the trailer ends with an empty flag of hex '00'. Based on the above, the author's approach does not require file-to-file jumps inside the ZIP archive and calculating the end-length. The author's improved footer's logic is exportable to applications using signatures for file carving, in similar environments.

A drawback of the author's approach could theoretically appear when the end of a file fits a cluster exactly, laying at cluster's boundary, or when the end of a file borders non-zero slack in the same cluster. A heavily slacked environment could potentially cause an incorrect increase in the length of the file-trailer. The author performed some additional tests by filling volumes in NTFS v 3.1 with small (4096 bytes) files and writing new variable-size files to the volumes, without wiping first. In these circumstances, the slack of the sector appeared to be filled with hex '00'. It is believed that sector-slack does not exist in recent versions of MS Windows and the empty space at the end of sector is filled with hexes '00' [52, 53]. Therefore, for NTFS volumes the described drawback is unlikely to be encountered in practice. However, if an incorrect increase in the length of the trailer takes place, in the author's opinion it would not lead to the carved file failing the ZIP test (or to invalidation of signature), but an MD5 test would fail.

This script (1) recovers the data of files residing in contiguous clusters and has the following workflow:

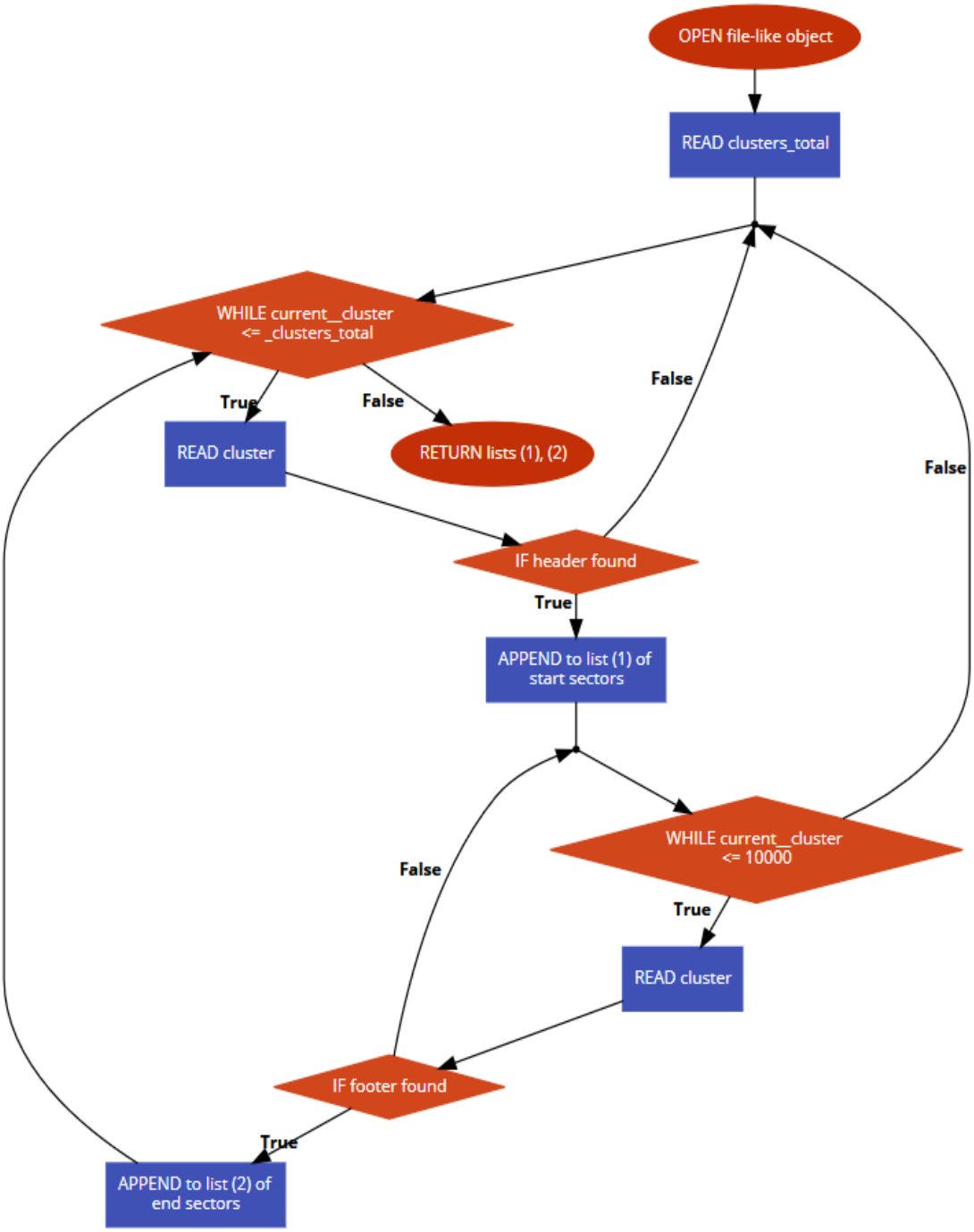

Figure 5[43]. Maximum file-size assisted header/footer based recovery of start and end sectors

Note that the workflow in Figure 5 is based on the header/footer carving technique discussed above and also relies on a maximum length fail-safe, administratively set to 10000 clusters. For testing purposes, the script (1) also entails the option of calculating the size of the trailer based on the 'Zip file comment' length value stored in Central Directory End Record's flag.

The script is thereafter applied to the raw image described above. The image in question has, in a deleted but recoverable state, 15 DSD-like containers (2 empty) and 14 other ZIP

[43] Assisted by Code2flow graphic engine, https://code2flow.com/app, retrieved on 31.03.2018.

containers, namely MS Office documents and ZIP archives. Of those files, 28 can be recovered by carving. The signatures applied are the following, provided here in Pythonic syntax for 'bytes' literal, an ASCII-hexadecimal regular expression friendly representation, with hexadecimal numbers designated by the prefix '\x':

1. Header standard ZIP, signature '^PK\x03\x04' [1, 48].
2. Header improved DSD-like container, signature '^PK\x03\x04(.|\s){26}mimetype'. This is a simplification of IANA registered 'magic numbers'.
3. Header official IANA DSD, signature '^PK\x03\x04(.|\s){26}mimetype(.|\s){12}vnd\.etsi\.asic\-e\+zip' [15]. Note that even though IANA mandates only 'PK' and leaves unspecified that the following bytes should be '\x03\x04', for practical readability the author includes these hexes in the signature. Hexes '\x03\x04' are also mandated by ZIP specifications [1, sec. 4.3.7] regarding Local File Header and can therefore be assumed to be present in actual archives. Also note the adjustment made in the syntax of the official header.
4. Header perfected DSD, signature '^PK\x03\x04(.|\s){26}mimetype(.|\s){0,28}(application\/vnd\.etsi\.asic\-e\+zip|K,\(\\\xc8\xc9LN,\xc9\xcc\xcf\xd3\/\xcbK\xd1K)'.
5. Footer standard ZIP, signature 'PK\x05\x06(.|\s){18}' [1, 48].
6. Footer improved ZIP, signature 'PK\x05\x06(.|\s){16}.*?(\x00{2}|.*[\w])'.

Note non-greedy repetition qualifier '*?' for the improved footer (6).

Table 6. Custom signature based script file carving start/end sectors recovery results for 15 sample DSDs and 13 recoverable other containers

Container type	**Standard ZIP header**	**Improved DSD header**	**IANA header**	**Perfected DSD header**
Non-DSD containers start/end sector pairs captured	13	0	0	0
DSD containers start/end sector pairs captured	15	15	11	15

As results above suggest the so-called "perfected DSD header" as well as "improved DSD header" signatures enable separating DSD containers from other ZIP like containers, in this particular case ZIP archives and MS Office documents, while ensuring carving of all DSD files.

To produce exact carved files the data needs to be extracted from the listed sectors with the end of the data matching the footer pattern. Footer signature selection bore no difference on the results in this particular case, if unzipping was applied to verify results. However, if an MD5 match was applied to compare the carved files with the originals, the improved footer signature provided for MD5 matches. The results of carving with the standard ZIP footer and with the improved footer measured in MD5 matches are depicted below.

Table 7. File carving with MD5 match results for 15 sample deleted DSDs

Test	Based on Script (1)		Recuva	Disk Drill	EnCase	Nuix	X-Ways	FTK
	Stand-ard ZIP footer	Im-proved footer						
MD5 match	11/15	15/15	15/15	15/15	0/15	0/15	15/15	6/15

Validation done at a later stage over larger set of samples will generally confirm the improved footer's capabilities and after further improvements, achieves complete MD5 matching. If only the standard ZIP header/footer is applied, a mix of different container types will be successfully carved with no satisfactory MD5 matching because of the wrong length of the trailer.

The same header and footer schemes as depicted in Figures 3 and 4 can be applied with forensic examination suites as well, sometimes requiring adjustments in the syntax compared to the one used in Python. For the header signature, the following expressions were tested:

EnCase 8:

'PK\x03\x04.{26,26}mimetype.{0,28}(application\/vnd\.etsi\.asic\-e\+zip)|(\x4B\x2C\x28\xC8\xC9\x4C\x4E\x2C\xC9\xCC\xCF\xD3\x2F\xCB\x4B\xD1\x4B)'

X-Ways 19.5 and FTK Imager 3.4:

'PK\x03\x04.{26}mimetype.{0,28}(application/vnd\.etsi\.asic\-e\+zip|K,\(\xc8\xc9LN,\xc9\xcc\xcf\xd3\/\xcbK\xd1K)'

For the footer signature, the following expressions were successful in correctly identifying the end of the file:

EnCase 8:

'PK\x05\x06.{16,16}(\x00{2,2}|.{2,2}[-~]*)'

X-Ways 19.5:

'PK\x05\x06.{16}(\x00{2}|.{2}[-_0-9a-zA-Z]{0,512})'

FTK Imager 3.4:

'PK\x05\x06.{16}(\x00{2}|.{2}.*?[-~]+)'

Note that in case of X-Ways the author administratively set the maximum distance of the footer to '512'. In some cases, the character class identified as alphanumeric '\w' in Python was replaced with a custom character class '[-~]', the meaning of which is better explained later in this thesis.

The author tested the regular expressions with the tools listed above in manual search mode for EnCase and FTK Imager and they produced the correct headers and footers for 15 DSD files in the set of 28 deleted but recoverable files. For X-Ways, the expressions were tested

in the automatic carving mode resulting in 15 files carved with MD5 matches. The conditions and the results of testing with the relevant tools are summarised in the table below.

Table 8. Detection of sample DSDs with certain tools based on the perfected DSD header and improved footer

Type	EnCase	X-Ways	FTK Imager
Testing method	Manual, "Raw Search All"	Automatic	Manual, "Find"
Options	GREP	See notes below	ANSI, Regular Expression
Header and footer pairs correctly detected	15	15	15

To test the header and footer signatures in X-Ways automated search and carving mode, a custom 'DSD Archive' file-type was created in the X-Ways' configuration file 'File Type Signatures Search.txt' with the above header and footer expressions and with the new extension of '.bdoc'. To account for the sector/cluster ratio of the media "Search all at sector boundaries" carving option was enabled[44]. As result, X-Ways produced 15 '.bdoc' files with matching MD5 hashes. Note that as the results summarized in Table 5 suggest, X-Ways should be perfectly capable of producing precise matches with its own algorithm no. 14 for the detection of the correct length of the file footer. The author thereafter tested file carving in X-Ways with the author's proposed header signature and with the X-Ways' proprietary footer algorithm no. 14. This test also resulted in successful recovery of 15 DSD containers, which matched the originals' MD5 hashes. This could suggest that to enable pin-pointed carving of DSD file-type with this particular tool, it is sufficient to add the custom, author's proposed header alone and retain the use of the proprietary footer algorithm. However, later tests performed over larger set of deleted files suggest that the author's footer signature is preferable to the algorithm in question.

For a second planned test, keyword searches were performed. Keyword searches were conducted over files in folders (for MS Windows) and over a forensic image containing intact files and an intact volume. For this purpose, the following keywords were selected (in table below).

Table 9. Keywords

Keyword	Original containers (number) and explanation	Layer
UUKKIVI	DSD (2) - signer's name	ASN.1 (X.509 certificate in signatures1.xml)
37501110300	DSD (9) - signer's personal code	ASN.1 (X.509 certificate in signatures0.xml)
Signator	DSD (5) - signer-added field	XML in signatures0.xml
Tuttavad	DSD (11) – document to be signed ZIP (1), DOCX (1)	ZIP
Outage	Sample DOCX (1) from the Internet	ZIP
prohibitively	Sample ZIP (1) from the Internet	ZIP

[44] Instructions provided in 'README.md' at https://github.com/raul-nugis/bdoc_carver, 19.05.2018.

Unlike the file carving exercise, keyword searches in forensic examination suites and MS Windows were run over the files or over an image of the media with the files intact. In this way the results were not dependent on the file recovery capabilities of a particular tool. Note that compound file mounting / expanding was enabled in all tools' search or indexing options, if available, to ensure that the tools would fully access zipped data. Searches were conducted over the whole image, not only over the selected files, with the exception of MS Windows where it was over files in a folder. The keywords selected and the results of the searches with different tools are depicted in a table below.

Table 10. Search results

Keyword	**Layer**	**Found (one per file) by a tool / present in original file**				
		X-Ways	EnCase	Nuix	FTK	MS Win10
UUKKIVI	ASN.1	0/2	0/2	0/2	0/2	0/2
37501110300	ASN.1	0/9	0/9	0/9	0/9	0/9
Signator	XML	5/5	5/5	3/5	5/5	5/5
Tuttavad	ZIP	13/13	13/13	13/13	13/13	13/13
Outage	ZIP	1/1	1/1	1/1	1/1	1/1
prohibitively	ZIP	1/1	1/1	1/1	1/1	1/1

The author then created a script (2)[45] based on an algorithm for retrieval of the attributive data described above. This script also incorporates the carving-based recovery already described and implemented in script (1). Applied to a forensic image as in the current case, this script (2) remains agnostic to the file state, reading data from both deleted and non-deleted files. If the signature based on the ZIP file header is applied, the script (2) is capable of reading classic ZIP files, even though no DSD specific contents can be recovered. If the signature based on the standard ZIP footer is substituted with the "improved footer" as described above, the script (2) should carve with MD5 matches of the original files. For file-like objects matching DSD-specific headers, the script (2) will also extract attributive data. The script makes use of existent Python libraries listed in the figure below, capable of parsing ZIP, XML and Base64-ASN.1 encoded X.509 certificate data.

[45] https://github.com/raul-nugis/bdoc_finder, 18.03.2018.

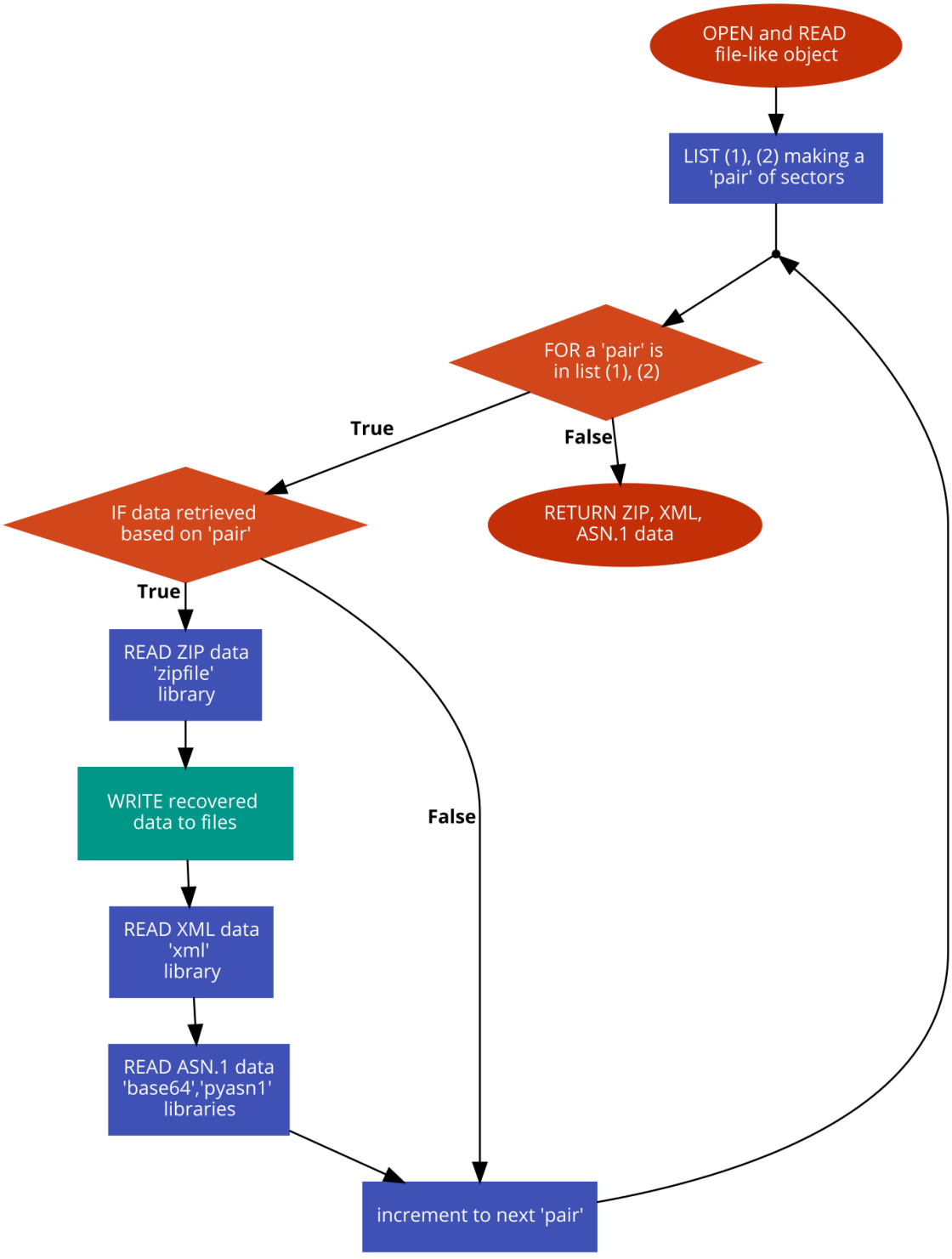

Figure 6. Extraction of attributive data from DSD combined with file carving

Note that in Figure 6 the script (1) for header/footer based carving with a maximum file-size limit as depicted in Figure 5 is applied in the first stage, i.e. in the rectangular box with the process description of "*LIST (1), (2) making a 'pair' of sectors*". Script (2) uses additional validation of data being carved based on the structure of DSD at the stage of unzipping of the contents.

This workflow produces attributive data identified in the observational stage, including extra and comment fields from the ZIP layer as well as meaningful XML data about the signer and signing time. Finally, names and personal codes of persons who signed the documents, as well as time-responses are retrieved from ASN.1 encoded data. This script does not recover data of the signed documents, of which all tested forensic tools as well as MS Windows indexing features proved to be fully capable. As applied to the test image containing deleted sample containers, this script was able to recover all of the keyword related data from the ASN.1 layer of the DSD containers and recover all containers with MD5 matches.

Table 11. Occurrences of search keywords in data extracted from the image with 15 deleted DSD files using script (2)

Keyword	**Script (2), occurrences found / occurrences known to exist**	**Container (recovered)**
UUKKIVI	2/2	32080_32144_Deleted and formatted volume.001.bdoc 32144_32208_Deleted and formatted volume.001.bdoc
37501110300	9/9	32232_32304_Deleted and formatted volume.001.bdoc 32304_32376_Deleted and formatted volume.001.bdoc 32376_32448_Deleted and formatted volume.001.bdoc 32448_32520_Deleted and formatted volume.001.bdoc 32520_32592_Deleted and formatted volume.001.bdoc 32592_32664_Deleted and formatted volume.001.bdoc 32776_32841_Deleted and formatted volume.001.bdoc 32848_32913_Deleted and formatted volume.001.bdoc 32920_32985_Deleted and formatted volume.001.bdoc

The script has 804 lines and 630 SLOC according to GitHub stats (21.05.2018).

7.2 Large-Scale Testing

An examination of sample containers demonstrated a variety of approaches purposely or unknowingly undertaken by developers of signing applications, with seemingly little consideration towards the standards' part regarding Local File Header. This makes the first exercise of this thesis, which is searching for a surgically precise file carving signature, more complicated. It is also reasonable to assume that the workflow proposed for retrieval of forensically valuable attributive data as based on the few available samples could benefit from further testing. Based on these considerations it would be suitable to test findings on a larger scale over many DSD files.

For this purpose, a number of different DSDs will need to be obtained. As noted above, SK ID Solutions provides only 4 test ID cards, setting certain limitations on their use. A number of additional samples can be obtained from a public registry of a government body in Estonia. Download can be facilitated with Web Browser automation. For these purposes the author created a web-scraping script based on the 'Selenium' library[46] v 3.1.0 for Python, in tandem with Firefox v 56.0 and Geckodriver v 0.19.0. The Selenium library was chosen because, in the author's experience, public document registries tend to rely on Web pages

[46] For details see http://selenium-python.readthedocs.io, retrieved on 19.03.2018.

with dynamically generated content, so is the case for this particular body's open registry. The registry in question is http://dokumendihaldus.terviseamet.ee/default.aspx.

The script is multi-stage. In the first stage the script loads the Web page, enters the search word 'bdoc' and presses the "Otsi" (Estonian for "Search") button. Thereafter the script, after a new search result page is successfully loaded, presses the "Järgmine leht" (Estonian for "Next page") button. The script collects links to possible BDOC documents by looping while the size of the search result page remains above a certain threshold, while at the same time changing the current search result state numeration at each page load.

After creating a list of links to possible files, a separate script downloads the files from the links collected in the first stage. This next stage script uses 'requests' v 2.13.0 and 'bs4' v 0.0.1 Python libraries, which are capable of the recovery of contents and files from static Web pages. The 'bs4' library is applied to collect various registry entries about files while 'requests' is applied to download files. The process of scraping links and downloading document files, including meaningful registry entries about these documents, is extensively documented in CSV files and Pandas v 0.19.2 DataFrames. The scripts have 1110 lines and 832 SLOC (23.04.2018).

4099 links to document registration pages from between 2015 and 2018 containing 'bdoc' with possible attached DSDs were extracted and documented. Scraped registry entries suggested that these documents arrived into the registry on 2986 occasions via e-mail, 753 times a document management system delivered the document, in 341 cases snail mail was entrusted with this task and there were 9 cases of hand delivery.

As result of web-scraping for BDOC, ASICE, ZIP, DOCX, XLSX, PPTX, ODF and ODT containers, the following actual files were downloaded: 3862 BDOC, 11 ASICE, 0 ZIP, 160 DOCX, 26 XLSX, 0 PPTX, 0 ODF, 32 ODT. The document registry did not contain accessible EPUB files. The following files were added, after being obtained randomly from the Internet: 10 EPUB[47] files, 5 ODF and 5 ODT documents to give a total set of 4111 files. As described above, EPUB and ODF/ODT files are considered, structurally, the closest containers to DSD containers, according to the referred ETSI standard.

The files were placed on the same hard drive (wiped) on a small NTFS volume of 409600 sectors and forensic images were acquired. As result of the application of the script (2) based on a perfected DSD header and improved footer (see remarks below) signatures 3873 DSD-like files were recovered, of which 4 DSD-like files were unsigned BDOC containers. 32 DSDs in the set also appeared to have 40 other DSDs, stored inside. Out of 3873 recovered containers, 3873 matched the originals based on an MD5 match. No other ZIP or document was recovered thus satisfying the objectives of pin-point recovery of DSD only.

During file carving the initially proposed file header signature was further changed to '^PK\x03\x04(.|\s){26}mimetype(.|\s){0,36}(application\/vnd\.etsi\.asic\-e\+zip|K,\(\\\xc8\xc9LN,\xc9\xcc\xcf\xd3\/\xcbK\xd1K)' with the change highlighted in colour. This is because of the varying length of 'Extra field' in Local File Header. The length of this optional field is set to a two-byte long flag [1, 36]. It is reasonable to assume that this field can be even longer because the highest possible value of two bytes is 'FF FF', which equals decimal 65535. In such cases, the highlighted number in the proposed perfected DSD header signature must be further increased.

During file carving the initially proposed signature for improved footer (look for option (6) in the corresponding section) was further changed to

[47] Freely given e-books from https://www.epubbooks.com, retrieved on 19.03.2018.

‘PK\x05\x06\x00\x00(.|\s){14}.*?(\x00{2}|.*[\w:])’ with changes highlighted in colour. The initially proposed footer signature would fail in producing correct MD5 matches in 9 containers out of 3873. Apart from failing the MD5 test this did not render those carved containers unusable in any other way. Adding white-space and a colon enabled full MD5 matching. In the author’s view it is reasonable to assume that the footer can in fact end not only with alphanumeric, but with any printable character, which means custom class ‘[-~]’ can be used in the footer, instead of ‘[\w:]’. This custom class ‘[-~]’ corresponds to all printable characters in the ASCII character table starting with space (hex ‘20’) and ending with tilde (hex ‘7E’).

Supplementing two hex-zero pairs after the standard-specified Central Directory End Record’s header as highlighted above was warranted because in one case a DSD contained a byte sequence matching the improved footer but present inside the zipped data and not at the end of file, thus generating a false positive and an invalid file-object. The footer could be improved in this fashion because, based on the referred ZIP specifications’ [1] paragraph 4.3.16, the first 3 pairs of bytes (6 bytes) are flags indicating placement of the ZIP archive over multiple disks. For a typical archive not placed on several disks and even more so for a DSD container, the footer’s beginning can be changed even further into ‘PK\x05\x06\x00\x00\x00\x00\x00\x00’.

Table 12. Results of large scale testing of DSD carving signatures

	Standard ZIP header and improved footer	**Improved DSD header and improved footer**	**IANA header and improved footer**	**Perfected DSD header and improved footer**
Total containers recovered	4111	3925	1453	3873
DSD containers recovered	3873	3873	1453	3873
Containers recovered ZIP test	4111	3925	1453	3873
DSD containers recovered MD5 match	3873	3873	1453	3873

The results depicted in Table 12 show that the perfected DSD header and improved footer signatures are successfully carving out the DSD files, separating them from the other containers present in the set. The signature for the header based on the IANA registered ‘magic numbers’, even with the previously identified syntax error compensated by the author, cannot satisfactorily recover the containers. The results also suggest that the standard ZIP header does not suit too well for pin-pointed recovery of DSDs, because of its broad scope related to generic ZIP. While the standard header signature indiscriminately carves out all ZIP containers including that of DOCX and XLSX, the improved header signature, even though eliminating MS Office Documents from the results, generates false positives by recognizing ODT, ODF and EPUB files (52 files in the set) as the same type of file with DSD containers. As described in the relevant chapter, the ASiC standard itself [14] points out that the corresponding file types are especially closely related to ASICE containers.

Additionally, the perfected DSD header and improved footer were tested in X-Ways v 19.5 automatic carving mode. The X-Ways friendly syntax for the header and footer described

in 7.1 was reinforced in the same way as the Python syntax described above. This resulted in the following expressions:

Header signature for X-Ways:

‘PK\x03\x04.{26}mimetype.{0,36}(application/vnd\.etsi\.asic\-e\+zip|K,\(\xc8\xc9LN,\xc9\xcc\xcf\xd3\/\xcbK\xd1K)’

Footer signature for X-Ways:

‘PK\x05\x06\x00\x00.{14}(\x00{2}|.{2}[-_0-9a-zA-Z]{0,512})’

The results of automated carving of the DSD files in X-Ways are depicted in the Table 13 below. This table describes the results of two tests. In the first test the author replaced the standard ZIP header with the author’s perfected DSD header in X-Ways configuration file’s ‘Header’ column. The value of the ‘Footer’ column was not replaced, which means that X-Ways’ proprietary algorithm no. 14 was used. In the second test the values in both ‘Header’ and ‘Footer’ columns were adopted for the use of the author’s signatures.

Table 13. Carving with X-Ways Forensics over large set of deleted files

	Perfected DSD header, X-Ways algorithm no. 14	**Perfected DSD header, improved footer**
DSD containers recovered	3873	3873
DSD containers recovered MD5 match	3868	3873

The results summarized in Table 13 indicate that X-Ways’ proprietary algorithm, which is meant for correct identification of the location and of the length of a ZIP file’s footer, has not achieved MD5 matches in 5 cases. Hex editor review reveals that 5 files in question have proper headers but appear to be “cut” well before the beginning of their ZIP Central Directory, amidst the data belonging to the embedded documents (DOCX, PDF), rendering the resulting carved containers practically unusable. Hex editor review undertaken by the author did not produce any obvious reason for this outcome.

In addition to carving, the script (2) also extracts all identified meaningful attributive data, including ASN.1 encoded data, from all recovered containers. During extraction, a single case appeared where the name of the person who signed a document was in ‘BMPString’ datatype and encoded in hex. Therefore, in the script extracting the data, decoding of ‘BMPString’ datatype was added.

Table 14. Extracted meaningful data

Extracted	**ZIP layer**	**XML layer**	**ASN.1 layer**
Records	15658	30492	63395
From files	3745	3869	3869

3745 of the containers appeared to have ZIP comment fields (‘ZIP file comment’ and/or ‘File comment’) or ‘Extra field’ filled. 1358 containers had 1409 unique file comments and 2478 containers had 2632 unique extra fields. ZIP file comments were filled in 118 containers. Containers were observed having several different comments or extra fields inserted. For example, in one case comment fields showed two of the three persons who signed a

document were using qdigidocclient v 3.13.4.1515 and Windows 8.1 (64 bit) and Windows 10 (64 bit) respectively while the third person was using Windows 10 (64 bit) with qdigidocclient v 3.13.3.1512, which means 3 different setups to sign a document.

'ZIP file comment' and 'File comment' suggest usage of different operating systems and applications.

Table 15. Central Directory comments' contents regarding operating systems and applications

Operating Systems or Application	Occurrences	
	File comment	ZIP file comment
Windows NT	26	0
Windows 7	496	0
Windows 8	99	0
Windows 10	310	0
Windows Server	73	14
Linux	305	106
Android	2	0
Mac OS	43	0
qdigidocclient	960	0
JDigiDoc	245	120
testtool	26	0
Amphora	40	0

'Extra field' suggests the most common flag is 'NTFS (Win9x/WinNT FileTimes)'. However, 'Extra field' values were extracted but not decoded.

The XML layer included 4442 signing times, indicating the recovered DSDs were signed between 2015-01-30 and 2018-03-13. The Signer's role related elements were filled on 2418 occasions.

Table 16. Selection of optional elements in XML signatures

Role	Occurrences	City	Occurrences	Province	Occurrences
direktor	1842	Tallinn	64	Ida-Virumaa	23
inspektor	230	Narva	8	Harjumaa	20
juht	216	Avinurme	8	Lääne-Virumaa	9
juhatus	118	Tartu	6	Valgamaa	6
vanem	100	Tõrva	6	Tartumaa	3
juhataja	79	Võsu	5	Raplamaa	1
minister	52	Kohtla-Järve	4	Pärnumaa	1
esimees	42	Pärnu	2	Jõgevamaa	1
linnapea	37				
spetsialist	14				
agent	1				

Note that in the table above, the number of individual occurrences for signer's roles exceeds the total number of occurrences of signer's roles. This is because multiple roles are possible. For example, there are 175 occurrences of "juhtivinspektor direktori ülesannetes", which combines "direktor" and "inspektor" in one role.

The results described in Table 15 and Table 16 make possible diverse types of findings relevant to the forensic field, and more.

For example, data extracted from the collected DSDs reveals that the Estonian Government's cabinet ministers, 8 of whom are present in the tested set of DSDs, make use of different setups for document signing. Among these setups, a certain version of an operating system's Kernel is extensively used. Because the use of this particular operating system is uncharacteristic for a typical office employee, and because the samples generated by the author using a web application also point to this operating system, it is possible that this practice of document signing relies on a document managing environment having a web-based platform and maintained in a unified fashion across many branches of the government. Findings like these can possibly be enhanced further by analysing time-stamps indicating prevalence of some systems during working hours.

Reviewing the ASN.1 layer script (2) produces 4442 signer's personal codes and 4440 names of signers as well as 2 names of corporate signatories and 8908 time-responses, aggregated in Table 17. The difference between the number of signatures and the number of time-responses in the analysed DSDs can be explained in the following fashion. An XML signature of a BDOC file contains an ASN.1 encoded 'EncapsulatedOCSPValue' object, which holds two time stamps with the same value. An XML signature of an ASICE file, on the other hand, additionally contains an 'EncapsulatedTimeStamp' ASN.1 encoded object, which adds two more time stamps with the total number of time stamps per signature double that of a BDOC.

Table 17. Some characteristics of data extracted from ASN.1 encoded objects

Role	**Occurrences**
Digital signatures	4440
Corporate signatures	2
Personal codes	4442
Time responses objects	8908
Signature missing	4
1 Signature per container	3339
2 Signatures per container	491
3 Signatures per container	30
4 Signatures per container	6
5 Signatures per container	1
Recursive[48] signature containers	40

The findings, on which Table 17 and Table 15 are based, likely point to possibilities of diverse types of analyses using the data extractable from DSDs and DSD-containing archives.

[48] Digitally signed document container included in other digitally signed document container.

For example, as shown in Table 17, four out of 3873 DSDs (roughly 0.1%) in the document registry from which the documents were initially scraped, are, format-wise, BDOC containers, but hold no signature. Combining this finding with the data about applications and operating systems aggregated in Table 15, the author suggests that in one case a certain person prepared a document for signing in MS Windows 7 Service Pack 1 (64 bit) using the qdigidocclient v 3.12.0.1448 application. It is difficult to ascertain how the document was not signed, but a possibility remains that the user just closed the application leaving an unsigned BDOC container. Based on web-scraped records and on the collected DSDs, the author can certainly suggest that the unsigned document was sent to the public authority by e-mail. The receiver of the document not only likely believed that the document was genuine, but also performed the tasks requested and subsequently issued their own document, briefly summarising the activities performed by the public body and the conclusions reached.

7.3 Results of Validation and Overall Findings

File carving exercises on both smaller and larger scales proved that a carving signature based on the header scheme depicted in Figure 3, having Pythonic regex representation '^PK\x03\x04(.|\s){26}mimetype(.|\s){0,36}(application\/vnd\.etsi\.asic\-e\+zip|K,\(\\\xc8\xc9LN,\xc9\xcc\xcf\xd3\/\xcbK\xd1K)' and applied at the beginning of a data unit on clustered media detects DSDs with high precision and separates them from any other tested container. The author believes that the header signature following the same logic can be better represented as '^PK\x03\x04(.|\s){26}mimetype(.|\s){0,VAR}(application\/vnd\.etsi\.asic\-e\+zip|K,\(\\\xc8\xc9LN,\xc9\xcc\xcf\xd3\/\xcbK\xd1K)' with variable '*VAR'* at least 36. The length of 'Extra field' present in Local File Header may be larger than 36 and, in such cases, *VAR* must be increased.

Carving tests also suggested that the signature based on the footer scheme depicted in Figure 4 having Pythonic regex representation 'PK\x05\x06\x00\x00(.|\s){14}.*?(\x00{2}|.*[\w:])' is capable of carving out the true length of a file in zero-slacked media. The author believes that a footer following the same logic can be better represented as 'PK\x05\x06\x00\x00(.|\s){14}.*?(\x00{2}|.*[-~])'.

Extraction of attributive data based on the algorithm proposed by the author proved successful in both smaller and larger scale tests. This included extraction of crucial ASN.1 encoded data from the embedded certificates and time-responses. Additional attributive data about the signers was extracted from the signature's XML elements. Detailed descriptions of operating systems and applications were frequently obtained from the containers' ZIP layer.

8 Conclusions

This thesis looked at digitally signed documents of the type most common in Estonia through the prism of a digital forensic examiner trying to unlock better ways to carve these files from raw data, as well as to learn what information they may possess, useful for investigating a wide range of crimes and infringements. These tasks were made easier by the existence of comprehensive documentation explaining the structure of digitally signed documents. These tasks were equally made harder because this documentation was created for different purposes, because the real-life occurrences of digitally signed documents did not always follow the standard, and because not one single mainstream forensic tool tested was able to separate DSDs from other ZIP containers, or penetrate all of their intricate layers.

As a result of the observation of sample documents, the author came up with a signature significantly more capable of pin-pointed carving of digitally signed documents than any other tested signature. The author mapped out data contained within digitally signed documents, which in the author's view possesses attributive properties, in other words helpful in identifying who signed the document and learning a little more about their environment. The author proposed an algorithm helpful in extracting such data.

Finally, this thesis put the author's ideas to the test, applying the carving signature and the algorithm to thousands of deleted real-life documents, collected through up-to-date web-scraping techniques. Within the environments used in the tests these ideas proved viable. The signature proposed in this thesis is capable of the pin-pointed recovery of virtually all containers, even when indiscriminately mixed together with structurally very close alternatives. Developed into a script, the algorithm for recovery of attributive data was capable of extracting this valuable information from every single document.

The author is hopeful that this thesis can be helpful to an examiner who does not have experience with digitally signed documents and wants a quick way of getting acquainted with this puzzling phenomenon.

References

[1] APPNOTE.TXT - .ZIP File Format Specification. Version 6.3.4, 2014. https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT (14.03.2018)

[2] Road Map for Digital Forensic Research. *The Digital Forensic Research Conference Technical Report*, 2001, p. 16. http://dfrws.org/sites/default/files/session-files/a_road_map_for_digital_forensic_research.pdf (15.03.2018)

[3] Garfinkel S. L. Carving contiguous and fragmented files with fast object validation. *Digital Investigation*, 2007, vol. 4, pp. 2-12.

[4] Digambar P.; Bhadran V.K.. Forensic Data Carving. *International Conference on Digital Forensics and Cyber Crime. ICDF2C 2010: Digital Forensics and Cyber Crime*, 2010, pp. 137-148.

[5] Ubelaker D. H. Forensic Science. John Wiley & Sons 2013, p. 224.

[6] Sepp O.; Priisalu J.; Suik K.. Eesti ID kaardi tehnoloogia. Lähteuuring. Küberneetika AS, 1998, pp. 2, 116. https://web.archive.org/web/20000823133250/http:/id.ee:80/idkaart/cyberaruanne.pdf (13.01.2018)

[7] Ansper A.; Buldas A.; Heiberg S.; Oit M.; Oone K.; Sepp O.; Villemson J.. Digitaalallkirja juurutamine riigiasutustes. Strateegiline plaan. 2001, pp. 20, 52. https://cyber.ee/uploads/2013/05/Digiallkiri_strat-plan-final.pdf (13.01.2018)

[8] Elektrondokumendi komisjoni seisukohad. https://web.archive.org/web/20010119123500/http://www.eik.ee:80/turve/eldok/skoht.htm (13.01.2018)

[9] Heinsoo E. Digitaalallkirja kasutamine kogub populaarsust. *Raamatupidaja.ee*, 2006. http://www.raamatupidaja.ee/uudised/2006/03/30/digitaalallkirja-kasutamine-kogub-populaarsust (29.01.2018)

[10] Priilinn K. Digiallkiri tavalise allkirjaga samaväärne. *Raamatupidaja.ee*, 2006. http://www.raamatupidaja.ee/uudised/2006/08/14/digiallkiri-tavalise-allkirjaga-samavaarne (20.01.2018)

[11] Report from the Commission to the European Parliament and the Council - Report on the operation of Directive 1999/93/EC on a Community framework for electronic signatures. *EUR-LEX*, 2006. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52006DC0120 (20.01.2018)

[12] Digitaalset allkirja kasutavate tööealiste (15-64-aastased) Euroopa liidu elanike osakaalu määramine 2015.aastal. *Ministry of Economic Affairs and Communications*, 2015. https://mkm.ee/sites/default/files/digitaalse_allkirja_kasutamise_osakaalu_uuringu_aruanne_printimiseks.pdf (23.02.2018)

[13] Mitašiūnas A.; Bykovskij A.. Lithuanian National Platform of Electronic Documents: Towards Cross-Border Interoperability. *eChallenges e-2015 Conference Proceedings*, 2015, p. 4.

[14] Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC). *European Telecommunications Standards Institute*, 2013, ETSI TS 102 918 V1.3.1 (2013-06), pp. 6, 8-10, 14-15, 20. http://www.etsi.org/deliver/etsi_ts/102900_102999/102918/01.03.01_60/ts_102918v010301p.pdf (14.03.2018)

[15] Ortega M. A. R. IANA registration for "vnd.etsi.asic-e+zip". *IANA*, 2013. https://www.iana.org/assignments/media-types/application/vnd.etsi.asic-e+zip (14.03.2018)

[16] Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC); Part 1: Building blocks and ASiC baseline containers. *European Telecommunications Standards Institute*, 2016, ETSI EN 319 162-1 V1.1.1 (2016-04), pp. 16, 25. http://www.etsi.org/deliver/etsi_en/319100_319199/31916201/01.01.01_60/en_31916201v010101p.pdf (15.03.2018)

[17] BDOC – FORMAT FOR DIGITAL SIGNATURES. BDOC2.1:2014, pp. 2-3, 11-12, 17. https://www.id.ee/public/bdoc-spec21-est.pdf (15.03.2018)

[18] BDOC – DIGITAALALLKIRJA VORMING. 2.1:2014. https://www.id.ee/public/bdoc-spec212-est.pdf (23.02.2018)

[19] Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES). *European Telecommunications Standards Institute*, 2010, ETSI TS 101 903 V1.4.2 (2012-10), pp. 10-15, 35, 43. http://www.etsi.org/deliver/etsi_ts%5C101900_101999%5C101903%5C01.04.02_60%5Cts_101903v010402p.pdf (15.03.2018)

[20] BDOC – FORMAT FOR DIGITAL SIGNATURES. BDOC 2.1:2013, p. 17. https://www.id.ee/public/bdoc-spec21.pdf (15.03.2018)

[21] BDOC – FORMAT FOR DIGITAL SIGNATURES. BDOC 2.0:2013, p. 17. https://www.sk.ee/repository/bdoc-spec20.pdf (15.03.2018)

[22] Libdigidocpp Programmer's Guide. http://open-eid.github.io/libdigidocpp/manual.html (15.03.2018)

[23] Volonino L. Electronic Evidence And Computer Forensics. *Communications of the Association for Information Systems*, 2003, vol. 12, art. 27, pp. 457-468. http://aisel.aisnet.org/cgi/viewcontent.cgi?article=3193&context=cais (15.03.2018)

[24] Chaski C. E. Who's At The Keyboard? Authorship Attribution in Digital Evidence Investigations. *International Journal of Digital Evidence*, 2005, vol. 4, iss. 1. http://www.flrchina.com/en/images/001/chaski_spring_05.pdf (15.03.2018)

[25] Cohen F. Digital Forensic Evidence Examination. Fred Cohen & Associates out of Livermore, 2013, pp. 34, 36.

[26] Buldas A.; Heero K.; Laud P.; Talviste R.; Willemson J.; Kullman K.; Seeba M.. Cryptographic algorithms lifecycle report 2016. Estonian Information System Authority, 2016. https://www.ria.ee/public/RIA/Cryptographic_Algorithms_Lifecycle_Report_2016.pdf (15.03.2018)

[27] Ansper A.; Buldas A.; Willemson J.; Seeba M.; Virunurm K.; Krüptograafiliste algoritmide elutsükli uuring 2017. Riigi Infosüsteemi Amet, 2018, pp. 9 - 12. https://www.ria.ee/public/RIA/kruptograafiliste_algoritmide_elutsukli_uuring_2017.pdf (15.03.2018)

[28] Kund O. ID-card tip from Czech scientists. Postimees, 07.09. 2017. https://news.postimees.ee/4236857/id-card-tip-from-czech-scientists (23.02.2018)

[29] Vines T. What Companies need to consider for e-Discovery. How Information Security Can Help the Organization Succeed. *SANS Institute InfoSec Reading Room*, 2015. https://www.sans.org/reading-room/whitepapers/legal/companies-e-discovery-36197 (15.03.2018)

[30] Bradford A. Empirical Evidence: A Definition. *Livescience.com*, 2017. https://www.livescience.com/21456-empirical-evidence-a-definition.html (07.03.2018)

[31] Marcus R. Only Yesterday: Reflections on Rulemaking Responses to E-Discovery. *Fordham Law Review*, 2004, vol. 73, iss. 1, p. 12.

[32] Väling K. Riigiprokuratuuri soovitused maksukuritegude kriminaalmenetluse läbiviimiseks. http://www.prokuratuur.ee/sites/www.prokuratuur.ee/files/elfinder/Maksukuritegude menetle mise soovitused.pdf (14.11.2017)

[33] Polley R. Digital Evidence Gathering in Dawn Raids – a Risk for the Company's Rights of Defence and Fundamental Rights. *20th St.Gallen International Competition Law Forum ICF*, 2013, pp. 1,7,10.

[34] Garfinkel S. L.; Migletz J. J.. New Xml-based files implications for forensics. *IEEE Security & Privacy*, vol. 7, iss. 2.

[35] Zhangjie F.; Xingming S.; Yuling L.; Li B.. Forensic investigation of OOXML format documents. *Digital Investigation*, 2011, vol. 8, iss. 1, pp. 48-55.

[36] Buchholz F. The structure of a PKZip file. https://users.cs.jmu.edu/buchhofp/forensics/formats/pkzip.html (08.03.2018)

[37] JDigiDoc Programmer's Guide, v 3.12. AS Sertifitseerimiskeskus, 2016, p. 74. https://www.id.ee/public/SK-JDD-PRG-GUIDE.pdf (12.03.2018)

[38] Tallinn District Court 15.04.2016 ruling in case no. 1-15-509, pp. 3, 6-7, 14-15. https://www.riigiteataja.ee/kohtulahendid/detailid.html?id=180104716 (31.03.2018)

[39] XML-Signature Syntax and Processing. W3C Recommendation 12 February 2002. The Internet Society and W3C, 2002. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/Overview.html (16.03.2018)

[40] Recommendation ITU-T X.660. Information technology – Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree. ITU, 2011, pp. 2-4, 8. https://www.itu.int/rec/T-REC-X.660-201107-I/en (31.03.2018)

[41] Cooper D.; Santesson S.; Farell S.; Boeyen S.; Housley R.; Polk W.. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Network Working Group, Request for Comments: 5280, pp. 13, 140. https://tools.ietf.org/html/rfc5280 (17.03.2018)

[42] Kröger K.; Creutzburg R.. A practical overview and comparison of certain commercial forensic software tools for processing large-scale digital investigations. *Proceedings of SPIE. The International Society for Optical Engineering*, 2013. https://www.researchgate.net/publication/258332973_A_practical_overview_and_comparison_of_certain_commercial_forensic_software_tools_for_processing_large-scale_digital_investigations (31.03.2018)

[43] Hernandez-Ardieta J. L.; Gonzales-Tablas A. I.; Fuentes J. M.; Ramos B.. A Taxonomy and Survey of Attacks on Digital Signature. *Computers & Security*, 2013, vol. 34, pp. 67-112.

[44] Viru District Court's Narva Courthouse's ruling in criminal case 1-18-825/5, p. 3. https://www.riigiteataja.ee/kohtulahendid/detailid.html?id=222662812 (22.03.2018)

[45] Veldre A. Turvanõrkus, oht, risk… RIA blog, 29.09.2017. https://blog.ria.ee/turvanorkus-oht-risk (22.03.2018)

[46] Recommendation ITU-T X.680. Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation. ITU-T. X.680 (07/2002), pp. vii, 5.

[47] Mikus A. N. An analysis of disc carving techniques. *Calhoun: The NPS Institutional Archive DSpace Repository*, 2005, pp. 1-4, 7-11, 33-36.

[48] Kessler G. C. File Signatures Table. https://www.garykessler.net/library/file_sigs.html (10.03.2018)

[49] Electronic Record. Society of American Archivists. https://www2.archivists.org/glossary/terms/e/electronic-record (23.02.2018)

[50] NIST. AU-10 Non-Repudiation. *NIST Special Publication 800-53* (Rev. 4). https://nvd.nist.gov/800-53/Rev4/control/AU-10 (14.03.2018)

[51] Mulazzani M.; Neuner S.; Kieseberg P.; Huber M.; Schrittwieser S.; Weippl E.. Quantifying Windows File Slack Size and Stability. 9th International Conference on Digital Forensics (DF), *IFIP Advances in Information and Communication Technology*, Springer 2013, pp.183-193. https://hal.inria.fr/hal-01460605/document (08.04.2018)

[52] Forensics: What is RAM Slack? https://whereismydata.wordpress.com/2009/04/26/forensics-what-is-ram-slack/ (08.04.2018)

[53] Larson S. P. Concerning File Slack. *ADFSL Conference on Digital Forensics, Security and Law*, 2009, p. 105. http://proceedings.adfsl.org/index.php/CDFSL/article/viewFile/113/110 (08.04.2018)

[54] Harichandran V. S.; Walnycky D.; Baggili I.; Breitinger F.. CuFA: A More Formal Definition for Digital Forensic Artifacts. *Digital Investigations*, 2016 vol. 18, pp. 125-137.

[55] Tamm M. ID-kaart murti lahti. RIA tõestas, et kära ID-kaardi turvanõrkuse pärast polnud asjata. *Eesti Päevaleht*, 19.04.2018. http://epl.delfi.ee/news/eesti/id-kaart-murti-lahti-ria-toestas-et-kara-id-kaardi-turvanorkuse-parast-polnud-asjata?id=81807683 (19.04.2018)

[56] Freed N.; Klensin J.. Media Type Specifications and Registration Procedures. Request for Comments: 4288. https://tools.ietf.org/html/rfc4288 (31.03.2018)

[57] Gül M.; Kugu E.. A Survey On Anti-Forensics Techniques. *Artificial Intelligence and Data Processing Symposium (IDAP)*, 2017.

[58] Character Classes and Bracket Expressions. GNU.org. https://www.gnu.org/software/grep/manual/html_node/Character-Classes-and-Bracket-Expressions.html (14.05.2018)

I. List of Annexes

1. Annex I. ZIP Structure and Hex Editor Based Review of Sample Containers ‘DMM.bdoc’ and ‘Bdoc21-TS.asice’
2. Annex II. Attributive XML Elements in Signature Part of Sample DSD Containers
3. Annex III. Attributive Data in ASN.1 Encoded Objects in DSD Sample Signatures
4. Annex IV. Carving, Keyword Searching, Extraction of Attributes of Sample Containers and Web-Scraped Containers
5. Annex V. Glossary

II. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Raul Nugis**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the Web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Forensic Data Properties of Digital Signature BDOC and ASiC-E Files on Classic Disk Drives,

supervised by Pavel Laptev, Raimundas Matulevičius

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **21.05.2018**